【新手学编程ABC丛书】

附赠 DVD ROM 光盘

- 全书19章PPT教学资源
- 311分钟实例精讲视频
- 全部315个程序源代码

Java编程新手自学手册

谭贞军 等编著

进入职场就是现在！



机械工业出版社
CHINA MACHINE PRESS



VISIT…

LANZAROTE

Caliente.COM

新手学编程 ABC 丛书

Java 编程新手自学手册

谭贞军　等编著



机 械 工 业 出 版 社

Java 是当今使用最为频繁的编程语言之一，一直在开发领域占据重要的地位。本书循序渐进、由浅入深地详细讲解了 Java 的核心技术，并通过具体实例的实现过程演练了各个知识点的具体使用流程。全书共 19 章，分为 4 篇。第 1～6 章是基础篇，逐一讲解了 Java 基础，Java 开发工具，Java 数据，字符串、运算符和表达式，假设语句，循环语句等 Java 开发所必须具备的基本知识。第 7～12 章是核心技术篇，逐一讲解了数组，面向对象，类，异常处理，I/O 与文件处理，线程等知识；第 13～17 章是提高篇，逐一讲解了网络与通信，AWT 开发窗体程序，窗口编程，数据库编程等知识。第 18、19 章是综合实战篇，分别通过画图板系统和网上书城系统的实现过程，讲解了 Java 语言在日常项目开发中的综合应用流程，并穿插介绍了各个模块的实现技巧。每篇最后为本篇的范例实战，通过实战演练帮助读者掌握本篇知识。全书采用故事性、趣味性相结合的对话讲解方式，并穿插了学习技巧和职场生存法则，引领读者全面掌握 Java。本书附有 1 张 DVD 光盘。

本书不但适用于 Java 的初学者，也适于有一定 Java 基础的读者，还可以作为有一定造诣的程序员的参考书。

图书在版编目（CIP）数据

Java 编程新手自学手册 / 谭贞军等编著. —北京：机械工业出版社，2012.4

（新手学编程 ABC 丛书）

ISBN 978-7-111-37937-9

Ⅰ. ①J…　Ⅱ. ①谭…　Ⅲ. ①Java 语言－程序设计　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2012）第 059507 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）

策划编辑：丁　诚

责任编辑：杨　硕

责任印制：杨　曦

保定市中画美凯印刷有限公司

2012 年 8 月第 1 版 • 第 1 次印刷

184mm×260mm • 34.75 印张 • 864 千字

0001－4000 册

标准书号：ISBN 978-7-111-37937-9

　　　　　ISBN 978-7-89433-496-1（光盘）

定价：89.90 元（含 1DVD）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
社 服 务 中 心：（010）88361066	
销 售 一 部：（010）68326294	门户网：http://www.cmpbook.com
销 售 二 部：（010）88379649	教材网：http://www.cmpedu.com
读者购书热线：（010）88379203	封面无防伪标均为盗版

丛 书 序

从《杜拉拉升职记》谈起

近年来，职场小说备受青睐，李可老师的《杜拉拉升职记》更是受到广大读者的喜爱，还被搬上了银幕。对许多人来说，职场生涯占据了整个人生的很大一部分时间，怎样才能在职场中如鱼得水，是人们必须认真思考的重要问题。即将走上程序员岗位的读者朋友们，请自问是否已经对未来的职场生涯胸有成竹？

本丛书不但可以帮助初学者提前演练职场生活，而且对在职人员也有借鉴意义。技术方面的知识就不用再多说了，每一页所包含的内容都是作者多年来的技术结晶。阅读本丛书后，希望读者们不但能学到编程技术，而且能够提前体验到职场中的一些常见场景，为将来的职场生涯做一些准备。希望本书能为读者们解惑，也希望能激励读者们在这个行业继续奋斗下去，迎接大家的将是明媚的阳光。

程序员的各个阶段

按照掌握技术的熟练程度来划分程序员的不同阶段，可以大体分为 5 个阶段。

1）初学者：处在此阶段的可能是一名在校学生，可能是应届毕业生，也可能是准备从其他行业向编程行业转行的人员。其共同特点是刚开始学习编程知识，对每一个知识点都充满了好奇，对未来充满期望。

2）菜鸟：这里的菜鸟可能是技术菜鸟，也可能是职场菜鸟。特点是某项技术的基本知识已经学习完毕，但是没有经过项目的洗礼，尚需要实战演练来磨练。这个阶段一般指处于试用期或者刚刚从事程序员工作的人员。

3）初级程序员：对项目开发的基本流程有了初步的认识，并通过工作实战演练了自己的技术。此阶段处于进一步与同事、上级、下级和客户交流的摸索阶段，也是逐渐融入职场的一个阶段。

4）高级程序员：开发经验丰富，技术扎实，对同事关系、上下级关系和客户关系已经如鱼得水，也是事业发展的瓶颈阶段。此阶段的程序员在职场中一般是软件高级工程师。

5）资深程序员：技术实力和人脉关系俱佳，一个项目任务能如探囊取物般轻松完成。但是也对自己的未来充满思索，想寻求待遇更好的职位，会考虑跳槽，也会考虑创业。为了表述得更加直观，下面通过一幅图来展示程序员的成长历程。

本丛书书目

根据综合考虑分析，本丛书首批书目如下。

C 语言编程新手自学手册

C#编程新手自学手册

Visual C++编程新手自学手册

Java Web 编程新手自学手册

Java 编程新手自学手册

PHP 编程新手自学手册

编程算法新手自学手册

技术菜鸟或职场菜鸟。特点是某项技术的基本知识已经学习完毕，但是没有经过项目的洗礼，尚需要实战演练来磨练。



处在此阶段的可能是一名在校学生，可能是应届毕业生，也可能是准备从其他行业向编程行业转行的人员。其共同特点是刚开始学习编程知识，对每一个知识点都充满了好奇，对未来充满期望。



对项目开发的基本流程有了自己的认识，并通过工作实战演练了初步的技术。此阶段处于进一步与同事、上级、下级和客户交流的摸索阶段，也是融入职场的一个阶段。



开发经验丰富，技术扎实，对同事关系、上下级关系和客户关系已经如鱼得水，但也是事业进一步发展的瓶颈阶段。



技术实力和人脉关系俱佳，一个项目任务能如探囊取物般轻松完成。对自己的未来充满思索，想寻求待遇更好的职位。会考虑跳槽，也会考虑创业。

致读者

学习程序开发之路是充满挑战之路，也是充满乐趣之路，这条路没有捷径可走。梦想像《天龙八部》中虚竹那样轻松获得一甲子功力，是不现实的。读者们要想真正学好编程，需要付出辛苦的汗水。根据笔者的亲身体会，替读者总结出 3 条学习编程的建议。

（1）培养兴趣

无论做什么事情，只要有了兴趣，就喜欢花费时间去做它。只要喜欢感受那调试成功的喜悦，就说明已经对编程产生了兴趣。这种喜悦会使自己更加喜欢编程，会带来成就感。闲暇时刻建议多去专业编程论坛逛一逛，灌灌水。论坛里的朋友们不但能帮助自己解决问题，而且还能带来其他非技术性的收获。

（2）脚踏实地

欲速则不达，学编程切忌有浮躁的心态。很多初学者刚学会了基本语法知识，调试成功了几段代码，就迫不及待大声宣布："我精通××语言了"。但是当遇到问题之后才发现，自己学到的只是九牛一毛。常说"书山有路勤为径，学海无涯苦作舟"，是很有道理的。

（3）多实践

程序开发很强调实践动手能力，所以实践就变得尤为重要。有前辈高人认为，学习编程的秘诀是"编程、编程、再编程，练习，练习，再练习"，笔者深表赞同。学编程不仅要多实践，而且要快实践。在看书的时候，不要等到完全理解了才动手，而是应该在看书的同时敲代码，程序运行的各种情况可以让自己更快、更牢固地掌握知识点。

我们的服务邮箱是 150649826@qq.com，读者在阅读本丛书时，如果发现错误或遇到问题，可以发送电子邮件及时与我们联系，我们会尽快给予答复。

丛书编委会

前　　言

Java 语言的重要性

Java 语言自正式发布起，经历了初生、成长和壮大的阶段，如今已经成为 IT 领域里的主流编程语言。Java 应用之所以如此广泛，要归功于它的以下特点：

（1）面向对象

Java 自诞生之时就设计成面向对象的语言。在 Java 眼里，一切都是对象，桌子和板凳是对象，花草树木和飞禽走兽是对象，异常和错误也是对象。

（2）跨平台

不管是 Windows 平台还是 UNIX 平台或其他平台，Java 程序都适用。Java 编译器把 Java 源程序编译成与操作系统平台无关的字节码指令。只要安装了 Java 虚拟机，Java 程序就可在任意的操作系统上运行。Java 程序中的字节码指令由 Java 虚拟机来执行，Java 虚拟机的解释器解析字节码，对它进行转换，使之能够在不同的操作系统平台上运行。

（3）直接支持分布式的网络应用

假定网络中有两台主机，分别运行着不同的 Java 程序，运用 Java 套接字技术，这两个 Java 程序就能顺利地进行远程通信。

（4）安全性和健壮性

Java 致力于检查程序在编译和运行时的错误，奉行“错误发现和纠正得越早，造成的损失就越小”的原则，做到防患于未然。Java 还支持自动内存管理，这不但为程序员减轻了负担，也减少了程序员犯错的机会，同时还减少了内存出错的可能性。

本书的特色

本书的最大特色是以作者学习历程的日记为主线，用一问一答的模式引出知识点。本书讲解了作者从一个入校新生学习编程开始，到顺利毕业并进入职场的成长历程。作者用日记的格式全程记录了过去的学习历程，日记和学习、职场息息相关，也和知识点紧密相连，使读者在阅读后有豁然开朗的感觉。

（1）成长经历为主线，以项目为单位，每个项目是一个故事

作者用日记格式记录了过去的学习历程，从作者学生时代讲起，一直贯穿整个学习过程。以日记的方式记录下了学习过程中的点点滴滴，每个日记故事趣味和知识并重，情节引人入胜。

（2）理论加实践的学习模式

书中遵循了理论加实践的写作模式，在每个知识点讲解完毕之后，都会用一个具体实例来演练知识点的应用方法，且这些实例都具有代表性。

（3）揭示学习和职场经验

书中展示了一些职场中的规则和经验，逐一向读者展现了学习、应聘、同事关系、客户关系、上下级关系、跳槽、创业和升职的经验和体会，以此启发读者，帮助读者在职场中少走弯路。

（4）给读者以最大实惠

在配套光盘中不但有书中实例的源代码，而且有全程视频讲解的 PPT 素材。本书还免费赠送给读者几个典型应用案例，并为书中的实际案例都配备了详细的视频讲解。

本书的内容

第 1～6 章是基础篇，介绍了 Java 语言开发所必须具备的基本知识，逐一讲解 Java 基础、开发环境的搭建、Java 基本语法、假设语句、循环语句等领域的知识，并采用理论结合实例的方式对各个知识点进行了剖析。

第 1 章　Java 基础
第 2 章　Java 开发工具介绍
第 3 章　Java 中的数据
第 4 章　字符串、运算符和表达式
第 5 章　Java 中的假设语句
第 6 章　循环语句

第 7～12 章是核心技术篇，逐一讲解了数组、面向对象、类、异常处理、I/O 体系与文件处理、多线程等领域的基本知识，并采用理论结合实例的方式对各个知识点进行了剖析。

第 7 章　特殊数据——数组
第 8 章　Java 面向对象
第 9 章　类
第 10 章　异常处理
第 11 章　I/O 与文件处理
第 12 章　Java 线程

第 13～17 章是提高篇，逐一讲解了和网络接轨、AWT 界面开发、Swing 编程、Servlet、数据库编程等领域的知识，并采用理论结合实例的方式对各个知识点进行了剖析。

第 13 章　网络与通信
第 14 章　AWT 开发窗体程序
第 15 章　深入 Java 窗口编程
第 16 章　Java 和数据库
第 17 章　数据库编程

第 18、19 章是综合实战篇，分别通过了画图板系统和“众望书城”网上系统的实现过程，讲解了 Java 语言在日常项目开发中的综合应用流程，并穿插介绍了项目的实现技巧。

第 18 章　画图板
第 19 章　“众望书城”网上系统

本书的适用对象

初学编程的自学者
编程爱好者
大、中专院校的老师和学生
相关培训机构的老师和学员

毕业设计的学生　　初、中级程序开发人员
程序测试及维护人员　　参加实习的初级级程序员
在职程序员　　资深程序员

致谢

本书的主要编写人员有谭贞军、陈强、张兴建、王梦、管西京、张子言、朱万林、李强、周秀、王孟、陈德春、周涛、刘海洋、关立勋、孟娜、王石磊、徐亮、张储、蒋凯、扶松柏、唐凯、焦甜甜、张斌、杨国华、杨絮、张玲玲。感谢作者们的辛勤汗水，也特别感谢机械工业出版社的所有工作人员，从他们身上我们学到了严谨的工作态度。

读者服务

在编写本书的过程中，我们始终本着科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。再次声明，我们的服务邮箱是 150649826@qq.com，读者在阅读本书时，如果发现错误或遇到问题，可以发送电子邮件及时与我们联系，我们会尽快给予答复。

感谢您购买本书，希望本书能成为您编程路上的领航者，祝您阅读快乐！

编　者

目　　录

丛书序
前言

第一篇　基　础　篇

第 1 章　Java 基础 …… 1
1.1　认识 Java …… 2
1.2　Java 初步 …… 2
1.2.1　Java 的起源 …… 2
1.2.2　Java 语言的特点 …… 3
1.2.3　Java 的一些名词解释 …… 4
1.3　搭建开发环境 …… 5
1.3.1　获得 JDK …… 5
1.3.2　轻松安装 JDK …… 7
1.3.3　JDK 配置如此简单 …… 8
1.4　体会 Java 程序 …… 11
1.4.1　Java 输出“我喜欢你” …… 11
1.4.2　编译和运行 Java 文件 …… 14
1.5　疑难问题解析 …… 15
职场点拨——谈 Java 的重要性 …… 15
第 2 章　Java 开发工具介绍 …… 17
2.1　认识 Java 的开发工具 …… 17
2.2　Java 开发工具简介 …… 18
2.2.1　Eclipse 简介 …… 18
2.2.2　Netbeans 简介 …… 19
2.2.3　JBuilder 简介 …… 20
2.3　Eclipse 的获得与安装 …… 21
2.3.1　获得 Eclipse …… 21
2.3.2　新建一个 Eclipse 项目 …… 22
2.4　NetBeans 的使用 …… 27
2.4.1　下载 NetBeans …… 27
2.4.2　安装 NetBeans …… 29
2.4.3　使用 NetBeans 新建项目 …… 33
2.5　疑难问题解析 …… 35
职场点拨——学习 Java 的正确态度 …… 35

第 3 章　Java 数据……37
3.1　量……37
3.1.1　常量……37
3.1.2　变量……39
3.2　数据类型……43
3.2.1　简单数据类型值范围……43
3.2.2　字符型……44
3.2.3　整型……45
3.2.4　浮点型……46
3.2.5　布尔型……47
3.3　运算符……48
3.3.1　算术运算符……48
3.3.2　关系运算符和逻辑运算符……50
3.3.3　位运算符……53
3.3.4　条件运算符……53
3.4　标识符和关键字……54
3.5　疑难问题解析……55
职场点拨——不同的客户，不同的处理方式……55
第 4 章　字符串、运算符和表达式……56
4.1　再看运算符……56
4.1.1　算术运算符……57
4.1.2　关系运算符和逻辑运算符……62
4.1.3　位运算符……64
4.1.4　条件运算符……65
4.1.5　赋值运算符……66
4.2　表达式……67
4.2.1　什么是表达式……67
4.2.2　表达式的优先级……68
4.2.3　表达式的应用……68
4.3　字符串……70
4.3.1　字符串的初始化……70
4.3.2　String 类……71
4.3.3　StringBuffer 类……77
4.4　疑难问题解析……79
职场点拨——提高你的职场生存能力……79
第 5 章　Java 中的假设语句……81
5.1　if 语句……81
5.1.1　if 控制语句……81
5.1.2　if 语句的延伸……83

5.1.3 多个条件判断的 if 语句 ……85
5.2 switch 语句 ……87
5.2.1 switch 语句的形式 ……87
5.2.2 switch 语句无 break ……90
5.2.3 case 没有执行语句 ……91
5.2.4 default 可以不在末尾 ……93
5.3 条件语句 ……94
5.3.1 正确使用 switch 语句 ……94
5.3.2 正确使用 if 语句 ……95
5.3.3 switch 语句的执行顺序 ……97
5.4 疑难问题解析 ……98
职场点拨——创业还是就业 ……98
第 6 章 循环语句 ……100
6.1 Java 循环语句 ……100
6.1.1 for 循环语句 ……100
6.1.2 while 循环语句 ……106
6.1.3 do…while 循环语句 ……109
6.2 跳转功能的实现 ……112
6.2.1 break 语句的应用 ……112
6.2.2 return 语句的应用 ……118
6.2.3 continue 跳转语句 ……120
6.2.4 轻松使用跳转语句 ……122
6.3 疑难问题解析 ……123
职场点拨——面试的准备 ……124
温故而知新——第一篇实战范例 ……125
范例 1 获得 JDK ……125
范例 2 配置运行环境 ……125
范例 3 安装 Java 的开发工具 ……126
范例 4 量、数据类型 ……126
范例 5 运算符 ……128
范例 6 表达式 ……129
范例 7 字符串 ……130
范例 8 if 语句 ……130
范例 9 switch 语句 ……131
范例 10 for 循环语句 ……132
范例 11 while 和 do…while 循环语句 ……133
范例 12 数组 ……134

第二篇　核心技术篇

第 7 章　特殊数据——数组 ……136
7.1　简单的一维数组 ……137
7.1.1　声明一维数组 ……137
7.1.2　创建一维数组 ……137
7.1.3　轻松初始化一维数组 ……139
7.2　二维数组 ……142
7.2.1　二维数组的声明 ……142
7.2.2　二维数组的创建 ……143
7.2.3　二维数组的初始化 ……145
7.3　多维数组 ……149
7.3.1　三维数组的声明 ……149
7.3.2　三维数组的创建 ……149
7.3.3　三维数组的初始化 ……149
7.4　对数组的操作 ……151
7.4.1　复制数组 ……151
7.4.2　比较数组 ……154
7.4.3　搜索数组中的元素 ……155
7.4.4　排序数组 ……156
7.4.5　填充数组 ……157
7.5　疑难问题解析 ……160
职场点拨——客户沟通之道 ……161
第 8 章　Java 面向对象 ……162
8.1　面向对象 ……163
8.1.1　面向对象的理念 ……163
8.1.2　面向对象的特点 ……163
8.2　面向对象的第一特征——类 ……164
8.2.1　如何编写一个类 ……164
8.2.2　特殊的方法——构造方法 ……164
8.2.3　一般的方法 ……167
8.3　属性和方法的修饰符 ……168
8.3.1　public 修饰符 ……168
8.3.2　private 修饰符 ……169
8.3.3　protected 修饰符 ……171
8.3.4　其他修饰符 ……171
8.4　this 的用法 ……173
8.5　类和对象的使用 ……175
8.5.1　创建和使用对象 ……175

8.5.2 使用静态变量和静态方法 ······176
8.6 特殊的类—抽象类 ······178
8.6.1 创建抽象类 ······178
8.6.2 抽象类的规则 ······180
8.7 软件包 ······181
8.7.1 定义软件包 ······181
8.7.2 在 Eclipse 中定义软件包 ······182
8.7.3 在程序中插入软件包 ······184
8.8 疑难问题解析 ······187
职场点拨——打造一个团队 ······187
第 9 章 类 ······189
9.1 类的继承 ······189
9.1.1 父类和子类 ······189
9.1.2 调用父类的构造方法 ······192
9.1.3 随意访问父类的属性和方法 ······195
9.1.4 多重次继承 ······197
9.2 重写和重载 ······200
9.2.1 重写 ······200
9.2.2 重载 ······205
9.2.3 重写与重载联合使用 ······208
9.3 接口 ······209
9.3.1 定义接口 ······209
9.3.2 接口里的量和方法 ······211
9.3.3 接口的实现 ······215
9.3.4 接口的引用 ······218
9.4 疑难问题解析 ······219
职场点拨——模块化设计的重要性 ······220
第 10 章 异常处理 ······222
10.1 什么是异常 ······222
10.1.1 认识异常 ······223
10.1.2 Java 提供的异常处理类 ······223
10.2 异常处理方式 ······224
10.2.1 使用 try…catch 处理异常 ······224
10.2.2 处理多个异常 ······226
10.2.3 在异常中使用 finally 关键字 ······227
10.3 将异常抛出 ······228
10.3.1 使用 throws 将异常抛出 ······228
10.3.2 使用 throw 将异常抛出 ······231
10.4 自定义异常 ······233

10.5 异常处理的陋习 ……236
10.5.1 丢弃异常 ……237
10.5.2 不指定具体的异常 ……238
10.5.3 占用资源不释放 ……238
10.5.4 不说明异常的详细信息 ……238
10.5.5 过于庞大的 try 块 ……239
10.5.6 输出数据不完整 ……239
10.6 疑难问题解析 ……240
职场点拨——不同老板的不同特点 ……241
第 11 章 I/O 与文件处理 ……242
11.1 Java I/O 简介 ……242
11.2 流 ……243
11.2.1 字节流 ……243
11.2.2 字符流 ……247
11.3 加快 I/O 操作效率 ……252
11.3.1 缓冲字节流 ……252
11.3.2 缓冲字符流 ……255
11.4 文件处理 ……257
11.4.1 文件类 ……257
11.4.2 使用文件类处理文件 ……258
11.5 疑难问题解析 ……260
职场点拨——可以做兼职 ……260
第 12 章 Java 线程 ……262
12.1 线程起步 ……262
12.1.1 线程与进程的理解 ……263
12.1.2 多线程的理解 ……263
12.2 创建线程 ……264
12.2.1 创建主线程 ……264
12.2.2 通过 runable 接口创建线程 ……265
12.2.3 通过 Thread 类创建线程 ……267
12.3 创建多线程 ……268
12.4 线程的优先级 ……271
12.5 控制线程 ……274
12.5.1 当前的线程等待 ……274
12.5.2 当前线程进入睡眠状态 ……278
12.5.3 当前线程做出让步 ……281
12.6 多线程同步 ……282
12.6.1 同步的重要性 ……282
12.6.2 轻松实现同步 ……284

12.6.3　什么是死锁 ……287
12.7　线程之间互相通信 ……289
12.8　疑难问题解析 ……292
职场点拨——揣测老板的弦外之音 ……292
温故而知新——第二篇实战范例 ……293
范例 1　类的继承 ……293
范例 2　接口的实现 ……295
范例 3　异常的处理 ……297
范例 4　读取文件的字符 ……298
范例 5　缓冲字节流 ……299
范例 6　深刻认识多线程 ……300
范例 7　认识网络编程 ……302

第三篇　提　高　篇

第 13 章　网络与通信 ……304
13.1　什么是网络通信 ……304
13.1.1　TCP/IP 协议 ……305
13.1.2　使用 URL 进行网络链接 ……305
13.1.3　编写 URL 程序常用的方法 ……306
13.2　网络编程初步 ……309
13.2.1　创建 Socket ……309
13.2.2　多个客户端连接 ……312
13.3　疑难问题解析 ……313
职场点拨——同事相处之道 ……314
第 14 章　AWT 开发窗体程序 ……315
14.1　什么是 AWT ……315
14.2　创建窗口 ……316
14.3　创建窗口组件 ……319
14.4　布局利器 ……321
14.4.1　布局利器 FlowLayout ……321
14.4.2　布局利器 BorderLayout ……323
14.4.3　布局利器 GridLayout ……326
14.4.4　布局利器 CardLayout ……329
14.4.5　布局利器 Null ……331
14.5　编写监听接口 ……334
14.5.1　窗口监听的接口 ……334
14.5.2　按钮监听的接口 ……336
14.5.3　文本框监听的接口 ……337
14.6　疑难问题解析 ……339

职场点拨——修炼“门面功夫” ……340
第 15 章　深入 Java 窗口编程……341
15.1　Swing 的开发步骤……341
15.2　创建窗口……342
15.2.1　JFrame 简介和方法……342
15.2.2　创建第一个 Swing 窗口……342
15.3　Icon 接口……344
15.4　添加组件……347
15.4.1　弹出式菜单……347
15.4.2　文本框……350
15.4.3　菜单……353
15.4.4　单选按钮……355
15.4.5　复选框按钮……358
15.4.6　列表框……359
15.4.7　选项卡……360
15.4.8　文本域……361
15.4.9　按钮……363
15.4.10　进度条……365
15.5　常用的布局管理器……366
15.5.1　不使用布局管理器……366
15.5.2　使用边界布局管理器……368
15.5.3　使用网格布局管理器……369
15.6　疑难问题解析……370
职场点拨——你准备找好工作吗……371
第 16 章　Java 和数据库……373
16.1　数据库的定义……373
16.2　操作 MySQL 数据库……374
16.3　MySQL 的安装……376
16.4　MySQL 的管理工具……379
16.4.1　创建数据库……379
16.4.2　创建表……380
16.4.3　输入记录……381
16.5　SQL Sever 很简单……383
16.5.1　创建数据库……383
16.5.2　创建表……385
16.5.3　创建记录……387
16.6　疑难问题解析……388
职场点拨——我有一颗创业心……388
第 17 章　数据库编程……390

17.1 SQL 操作 ······390
17.2 什么是 JDBC ······392
17.2.1 JDBC API ······392
17.2.2 JDBC 驱动类型 ······392
17.3 连接数据库 ······393
17.3.1 轻松连接 MySQL ······393
17.3.2 轻松连接 SQL Sever 2000 ······398
17.4 SQL 语句 ······403
17.4.1 新建数据库表 ······403
17.4.2 数据库查询语句 ······404
17.4.3 数据库操纵语句 ······406
17.5 疑难问题解析 ······408
职场点拨——谈加薪升职 ······409
温故而知新——第三篇实战范例 ······410
范例 1 顺序布局 ······410
范例 2 网格布局 ······411
范例 3 Swing 窗口（一） ······412
范例 4 Swing 窗口（二） ······414
范例 5 新建 MySQL 数据库 ······415

第四篇 综合实战篇

第 18 章 画图板 ······417
18.1 系统概述与预览 ······417
18.1.1 软件概述 ······417
18.1.2 软件预览 ······418
18.2 创建软件的准备 ······421
18.2.1 搜集素材 ······421
18.2.2 获得 Java API 手册 ······422
18.3 编程软件 ······422
18.3.1 创建一个类 ······422
18.3.2 菜单栏和标题栏的程序 ······423
18.3.3 保存文档的程序 ······426
18.3.4 界面的实现 ······430
18.3.5 调色盘的实现 ······436
18.3.6 中央画布的实现 ······443
18.3.7 输入字体的实现 ······446
18.3.8 打开以前文档的实现 ······448
18.3.9 实现其他功能 ······454
第 19 章 “众望书城”网上系统 ······460

19.1　效果展示……460
19.2　数据库设计……462
19.3　SQL Server 2000 JDBC 驱动……466
19.3.1　下载 JDBC 驱动……466
19.3.2　安装 JDBC 驱动……466
19.3.3　配置 JDBC 驱动……468
19.3.4　将 JDBC 驱动加载到项目中去……469
19.4　系统设计……470
19.4.1　登录窗口的编写……470
19.4.2　主窗口……472
19.4.3　商品信息的基本管理……479
19.4.4　进货信息管理……489
19.4.5　销售信息管理……498
19.4.6　库存管理……506
19.4.7　查询与统计……513
19.5　数据库模块的编程……517
温故而知新——第四篇实战范例……530
范例 1　编写记事本……530
范例 2　使用 Java 编写简易计算器……534

第一篇　基　础　篇

第 1 章　Java 基础

Java 是什么？Java 相比其他种类繁多的语言有什么特点？为何要选择 Java 语言？它能为人们的生活创造什么价值？

这些疑惑将在本章中为读者朋友一一解答，相信初次接触 Java 的初学者在学习本章之后，能够对 Java 有一个初步的了解。

本章主要内容如下：

- 认识 Java。
- 搭建 Java 开发环境。
- 体会 Java 程序。
- 职场点拨——看 Java 的重要性。

背景介绍

小菜，21 岁，大四学生，准备自学 Java。

Wisdom，小菜的表哥，资深软件工程师。

2008 年 X 月 X 日，天气阴

老师建议我选修 Java 语言。在过去的三年中，我一直在学习 C 语言和 C++语言，如今大四了才开始学习 Java，不知道对我以后的职场生涯有没有帮助……



一问一答

小菜："我以前学过 C 语言和 C++，现在学习 Java，还有必要吗？"

Wisdom："当然有必要，Java 是当今最主流的编程语言之一。在快速发展的今天，Java 广泛应用于各个领域，例如 Web 系统、桌面程序和手机应用等。建议你阅读本章最后的'谈 Java 的重要性'内容，你会得到一个明确的答案。"

1.1　认识 Java

在快速发展的今天，Java 广泛应用于各个领域，如手机游戏、银行系统等。Java 不只是一个程序，还是一个平台，一门语言。平时所说的 Java，通常指的是 Java 语言。本书主要讲解的就是 Java 语言。下面通过一个简单的游戏界面进入本章的开篇功能展示，图 1-1 所示为利用 Java 语言开发的一款打字游戏的界面。



图 1-1　打字游戏

1.2　Java 初步

Java 语言十分优秀，它给整个程序世界带来了巨大的影响，那么这门语言是怎么来的呢？它到底有什么魅力呢？通过学习本节内容，读者朋友将会初步了解 Java 的由来和特点。

1.2.1　Java 的起源

Sun Microsystems 公司（简称 Sun 公司）于 1995 年 5 月推出 Java 程序设计语言（以下简称 Java 语言），利用 Java 开发的程序实现了跨平台和动态 Web 两大特点，通过 Java 实现的 HotJava 浏览器（支持 Java applet）显示了 Java 语言无穷的魅力，可以说是 Java 推动了 Web 技术的迅速发展。自此，Java 开始被广大程序员接受。现如今，常用的浏览器都支持 Java Applet。

Java 平台由 Java 虚拟机（Java Virtual Machine，JVM）和 Java 应用编程接口（Application Programming Interface，API）构成。在硬件或操作系统平台上建立一个 Java 运行平台之后，就可以运行 Java 应用程序了。目前，Java 平台已经嵌入到了几乎所有的操作系统，Java 程序只需编译一次，就可以在各种系统环境中运行。而 Java 应用编程接口也从 1.1x 版发展到 1.2 版。目前常用的 Java 平台基于 Java1.6 版本，最新版本为 Java 1.7。如图 1-2 所示的是 Java 的标志。

Java 分为三个体系：Java ME（Java 2 Platform Micro Edition，Java 平台微型），Java SE（Java2 Platform Standard Edition，Java 平台标准版），Java EE（Java 2 Platform，Enterprise Edition，Java 平台企业版）。

2009 年 04 月 20 日，Oracle（甲骨文）公司宣布收购 Sun 公司，从此 Java 成为了 Oracle 系列产品之一。



图 1-2　Java 的标志

1.2.2　Java 语言的特点

Java 语言诞生于 1991 年，起初被称为 OAK 语言，是 Sun 公司为一些消费性电子产品设计的一个通用环境。他们最初的目的只是为了开发一种独立于平台的软件技术，在网络出现之前，OAK 一直默默无闻，甚至差点夭折，可以说真正改变了 OAK 命运的就是网络。在 Java 出现以前，Internet 上的信息内容都是一些乏味死板的 HTML 文档。这对于那些迷恋于 Web 浏览的人来说简直是不可容忍的。他们迫切希望能在 Web 中看到一些交互式的内容，开发人员也希望能够在 Web 上创建一类无须考虑软硬件平台就可以执行的应用程序，当然这些程序还要有足够的安全保障。对于用户的这种要求，传统的编程语言显得苍白无力。Sun 公司的工程师敏锐地察觉到了这一点，于是从 1994 年起，他们开始将 OAK 技术应用于 Web 上，并且开发出了 HotJava 的第一个版本。1995 年在 Sun 公司推出 Java 语言之后，全世界的目光都被这门神奇的语言所吸引。

Java 是一种简单的、面向对象的、分布式的、解释型的、健壮安全的、结构中立的、可移植的、性能优异和多线程的动态语言。现在的 Java 主要有如下特性：

（1）Java 语言是简单的

Java 语言的语法与 C 语言和 C++语言很接近，这使得大多数程序员可以轻松地学习和使用 Java。Java 不但不使用指针，还提供了自动的内存收集机制，使得程序员不必为内存管理而担忧。另一方面，Java 丢弃了一些 C++中很少使用的、很难理解的、令人迷惑的特性，如操作符重载、多继承、自动强制类型转换等，使整个 Java 语言更简洁，使用起来也更加方便快捷。

（2）Java 语言是面向对象的

Java 语言具有了类、接口和继承等特性，为了简单起见，只支持类之间的单继承，但支持接口之间的多继承，并支持类与接口之间的实现机制（关键字为 implements）。Java 语言全面支持动态绑定，而 C++ 语言只对虚函数使用动态绑定。

（3）Java 语言是分布式的

在基本的 Java 应用编程接口中有一个网络应用编程接口（Java.NET），它提供了用于网络应用编程的类库，库内包括 URL、URLConnection、Socket、ServerSocket 等类。这使得在 Java 中比在 C 或 C++中更容易建立网络连接，使网络编程更加简单。

（4）Java 语言是健壮的

Java 的强类型机制、异常处理、垃圾的自动收集等是 Java 程序健壮性的重要保证。对指针的丢弃是 Java 的明智选择。Java 的安全检查机制使得 Java 更具健壮性。

（5）Java 语言是安全的

Java 通常被用在网络环境中，为此，Java 特别提供了一个安全机制以防恶意代码的攻

击。Java 对通过网络下载的类具有一个安全防范机制，例如分配不同的名字空间以防其替代本地的同名类和字节代码检查，并提供安全管理机制允许 Java 应用设置安全哨兵。支持 Java 的浏览器还允许用户控制 Java 软件对本地系统的访问。各种措施的综合使用使 Java 程序的安全性得到了保证。

（6）Java 语言是可移植的

Java 在执行程序时，会自动将字节码转换为当前机器的机器码，所以程序开发人员无须考虑使用应用时的硬件条件和操作系统结构，用户也只需具备 Java 的运行平台，就可运行编译过的字节码。

（7）Java 语言是解释型的

Java 程序在 Java 平台上被编译为字节码格式， 然后可以在实现这个 Java 平台的任何系统中运行。在运行时，Java 平台中的 Java 解释器对这些字节码进行解释执行，执行过程中需要的类在连接阶段被载入到运行环境中。

（8）Java 是高性能的

与其他解释型的高级脚本语言相比，Java 的确是高性能的，而 Java 的运行速度随着 JIT（Just-In-Time）编译器技术的发展也越来越接近于 C++。

（9）Java 语言是多线程的

在 Java 语言中，线程是一种特殊的对象，它必须由 Thread 类或其子（孙）类来创建。通常使用如下两种方法来创建线程：

- 使用带参数的构方法 Thread（Runnable）实现 Runnable 接口的对象创建线程。
- 从 Thread 类派生出子类并重写 run 方法，使用该子类创建的对象即为线程。

因为 Thread 类已经实现了 Runnable 接口，所以任何一个线程均有它的 run 方法，而 run 方法中包含了线程所要运行的代码。线程的活动由一组方法控制。

Java 语言支持多个线程的同时执行，并提供多线程之间的同步机制（关键字为 synchronized）。使用不同的线程可以分别控制声音和图像，可以将多种处理融合在一起，在程序设计时只关心不同时候应该做什么处理，不必理解内部做何处理，这也提高了程序的动态交互性和实时性。

（10）Java 语言是动态的

Java 语言的目标之一是适应动态变化的环境。Java 程序需要的类能够动态地被载入到运行环境，也可以通过网络来载入所需要的类。

1.2.3 Java 的一些名词解释

在 Java 中有许多专业术语，这些专业术语通常是指 Java 语言下的某项技术或某个功能，下面将介绍常用的技术名词。

（1）JDBC

JDBC（Java Database Connectivity，Java 数据库连接）主要完成三件事：与数据库建立连接，传送 SQL（结构化查询语言）语句，最后执行 SQL 语句返回结果。JDBC 主要提供连接各种关系数据库的统一接口，可以为多种关系数据库提供统一访问，它由一组用 Java 语言编写的类和接口组成。JDBC 为工具和数据库的开发人员提供了一个标准的 API 接口，据此可以构建更高级的工具和接口，使数据库开发人员能够用纯 Java API 编写数据

库应用程序。

（2）EJB

EJB（Enterprise JavaBeans）使开发者可以方便地创建、部署和管理跨平台的基于组件的企业应用。

（3）Java RMI

Java RMI（Java Remote Method Invocation，Java 远程方法调用），主要用来开发分布式 Java 应用程序。一个 Java 对象的方法能被远程 Java 虚拟机调用。这样，远程方法激活可以发生在对等的两端，也可以发生在客户端和服务器之间，它使存储于不同地址空间的程序级对象之间彼此进行通信，实现远程对象之间的完美调用。

（4）Java IDL

Java IDL（Java Interface Definition Language）提供与 CORBA（Common Object Request Broker Architecture）的无缝的互操作性。

（5）JNDI

JNDI（Java Naming and Directory Interface，Java 命名和目录接口）提供了从 Java 平台到其他程序的无缝接口。这个接口屏蔽了企业网络所使用的各种命名和目录服务。

（6）JMAPI

JMAPI（Java Management API，Java 管理 API）为不同网络系统、网络和服务管理的开发提供一整套丰富的对象和方法。

（7）JMS

JMS（Java Message Service，Java 消息服务）提供企业消息服务，如可靠的消息队列、发布和订阅通信。

（8）JTS

JTS（Java transaction Service）提供存取事务处理资源的开放标准，这些事务处理资源包括事务处理应用程序、事务处理管理及监控。

（9）JavaBean

在 Java 技术中，除了上面的一些，值得关注的还有 JavaBean 技术，它是一个开放的标准的组件体系结构，它独立于平台，但使用 Java 语言，一个 JavaBean 是一个满足 JavaBean 规范的 Java 类，通常定义了一个现实世界的事物或概念。一个 JavaBean 的主要特征包括属性、方法和事件。

1.3　搭建开发环境

一个 Java 程序要正常进行编译和运行，用户就必须安装 JDK（Java Development Kit），然后再配置环境变量，下面将详细讲解这些知识点。

1.3.1　获得 JDK

JDK 是 Sun 公司针对 Java 开发员的产品。自从 Java 推出以来，JDK 已经成为使用最广泛的 Java SDK。JDK 是整个 Java 的核心，包括了 Java 运行环境、Java 工具和 Java 基础的类库。JDK 是开发和运行 Java 环境的基础，用户要对 Java 程序进行编译时，必须要获得对应

操作系统的 JDK，不然 Java 程序无法编译，获得 JDK 十分简单，其具体操作如下：

1）打开浏览器，在地址栏中输入 http://www.oracle.com/index.html，然后按〈Enter〉键浏览这个网站，如图 1–3 所示。

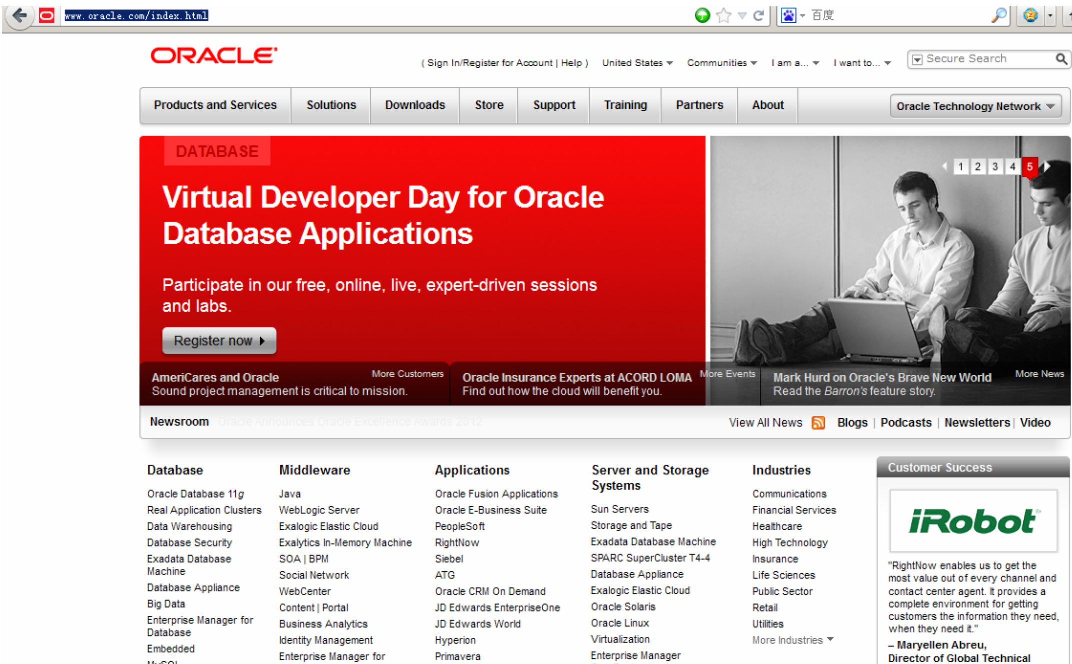


图 1–3　Sun 公司的下载主页

（2）单击顶部的“Download”链接，然后单击“Java Runtime Environment（JRE）”链接下载 JDK，如图 1–4 所示。



图 1–4　选择需要的 JDK

3）在弹出的新界面中单击“免费 Java 下载”按钮开始下载，如图 1-5 所示。



图 1-5　JDK 的下载页面

1.3.2　轻松安装 JDK

因为安装文件较大，下载 JDK 需要一定的时间。下载完成后，用户就可以双击安装文件进行安装，具体操作如下：

1）双击下载的“.exe”文件开始进行安装，将弹出“安装向导”对话框，在此单击“下一步”按钮。在弹出的“安装路径”对话框中选择文件的安装路径，如图 1-6 所示。

2）然后单击“下一步”按钮开始在安装路径解压缩下载的文件，如图 1-7 所示。



图 1-6　“许可协议”窗口

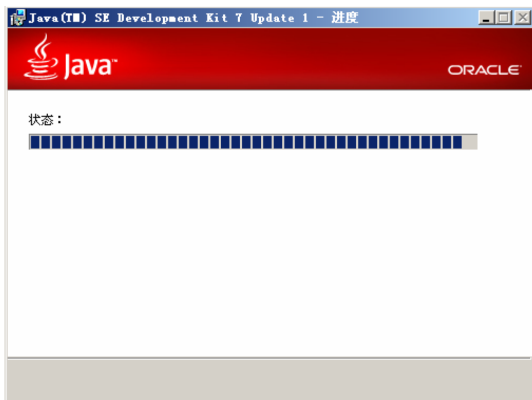


图 1-7　自定义安装

3）完成后弹出“目标文件夹”对话框，在此选择要安装的位置。如图 1-8 所示。

4）单击“下一步”按钮后开始正式安装，如图 1-9 所示。

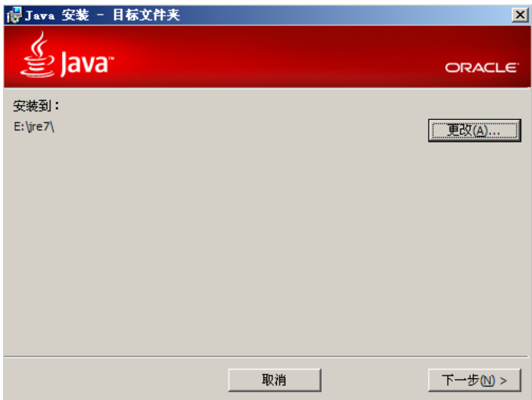




图 1-8　正在安装 JDK　　图 1-9　“目标文件夹”窗口

5）完成后弹出“完成”对话框，单击“完成”按钮后完成整个安装过程。如图 1-10 所示。

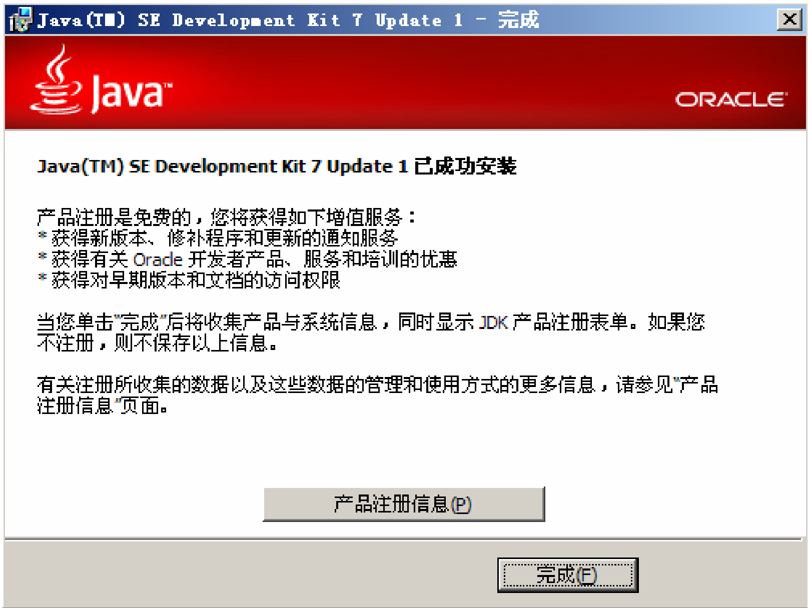


图 1-10　成功安装

1.3.3　JDK 配置如此简单

安装好 JDK 后，计算机并不能立即编译和运行 Java 程序，用户要想在环境中编译和运行 Java 源程序，还需要对计算机环境进行配置，配置的方法十分简单，具体操作如下：

1）在桌面上选中“计算机”图标，单击鼠标右键，在弹出的快捷菜单中，选择“属性”命令，打开“系统”窗口，如图 1-11 所示。

2）单击左侧任务栏中的“高级系统设置”超级链接，打开“系统属性”窗口，单击打开“高级”标签，如图 1-12 所示。



图 1-11　“系统”窗口

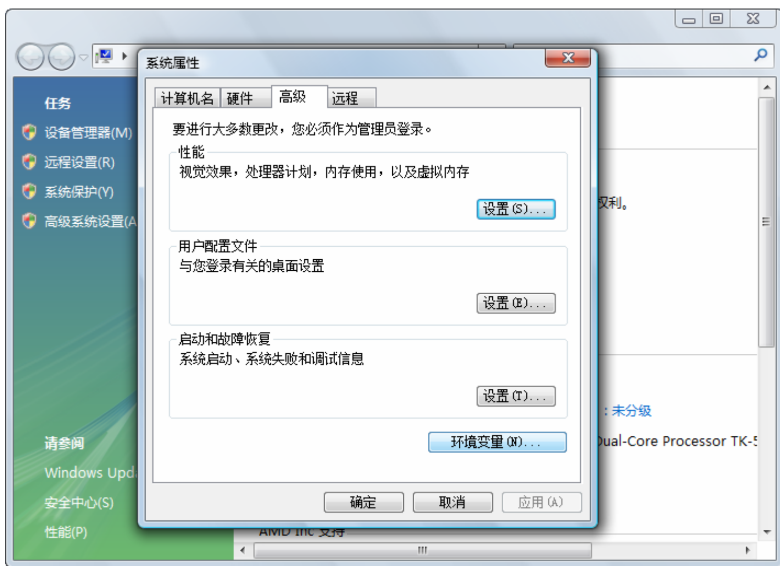


图 1-12　“系统属性”窗口

3）单击右下角的“环境变量”按钮，打开“环境变量”窗口，单击“新建”按钮对环境进行配置，如图 1-13 所示。

提示：在环境变量的配置过程中，会有用户变量和系统变量两种。若用户只针对当前登录的用户进行配置，则使用上面一栏“用户变量”。若用户针对整个系统的所有用户进行配置，则使用下面一栏“系统变量”。

4）弹出“新建用户变量”窗口后，在“变量名”文本框中输入“CLASSPATH”，然后在“变量值”文本框中输入 dt.jar、tools.jar 和 rt.jar 的文件路径，如图 1-14 所示。

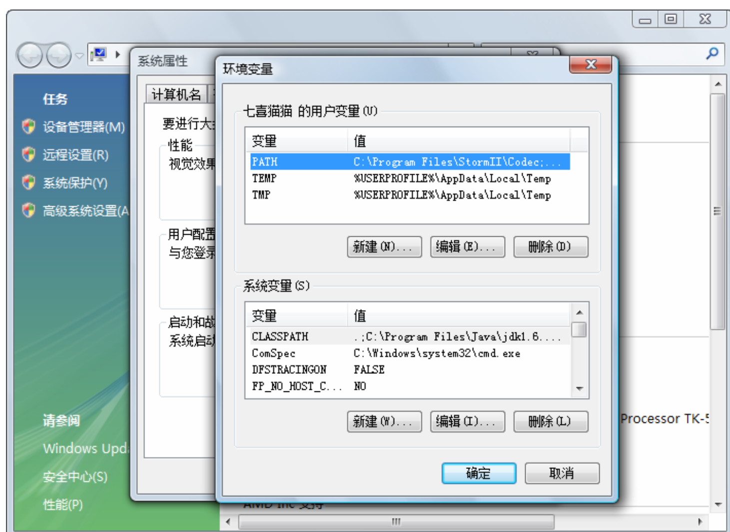


图 1-13 “环境变量”窗口

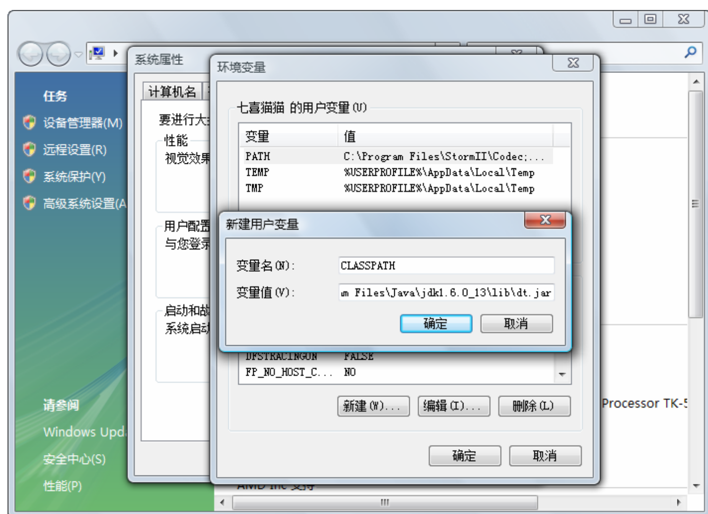


图 1-14 “新建用户变量”窗口

提示：读者安装的 JDK 可能跟笔者的不一样，但是具体配置方法都是大同小异。笔者将 JDK 安装在了默认目录下，所以在此设置 CLASSPATH 的变量值是“C:\Program Files\Java\jdk1.6.0_13\lib\tools.jar;C:\ProgramFiles\Java\jdk1.6.0_13\jre\lib\rt.jar;C:\Program Files\ Java\jdk1.6.0_13\lib\dt.jar;”。初学者一定要注意，必须在输入三个文件的路径前输入“;”。

5）设置完成后，单击“确定”按钮，然后单击“新建”按钮，打开“新建系统变量”窗口，在“变量名”文本框中输入“PATH”，在“变量值”文本框中输入 JDK 的 bin 路径和 jre 的 bin 文件夹的路径，如图 1-15 所示。

提示：本例是按照默认设置安装的，所以它的 PATH 的变量值是“C:\Program Files\Java\jdk1.6.0_13\bin; C:\Program Files\Java\jdk1.6.0_13\jre\bin;”。

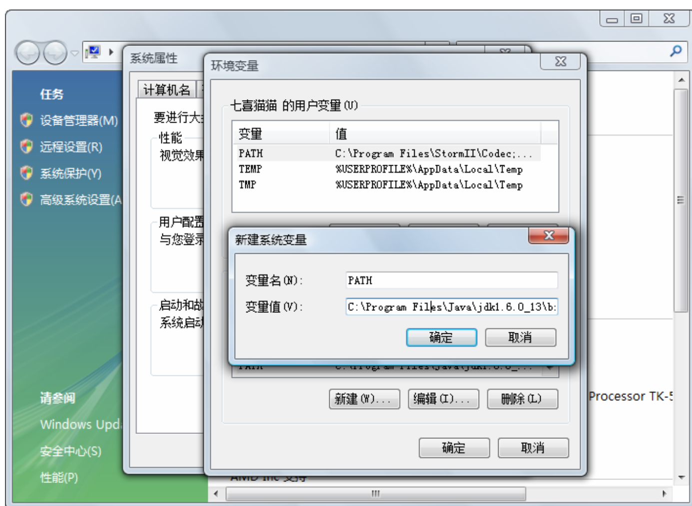


图 1-15　新建系统变量 PATH

6）单击“开始”菜单，选择“运行”命令，在文本框中输入“CMD”，按〈Enter〉键，然后输入“javac”，将会显示如图 1-16 所示的系统信息。

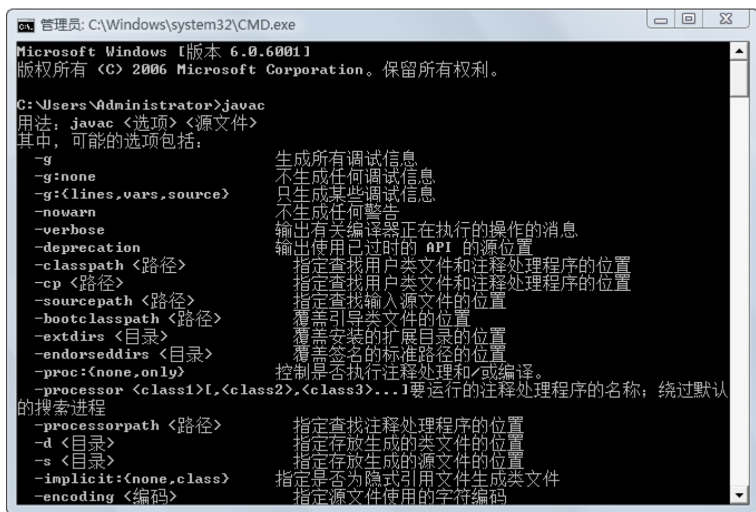


图 1-16　配置成功

1.4　体会 Java 程序

完成环境配置后，就可以书写 Java 程序了，那么应该如何书写 Java 程序呢？下面将详细讲解。

1.4.1　Java 输出“我喜欢你”

在安装并配置好 JDK 后，可以通过编写一个 JDK 程序测试 Java 的运行环境是否正确，

具体操作如下：

1）打开任意一个磁盘，在“组织”的下拉列表中，选择“文件夹和搜索选项”命令，如图 1-17 所示。

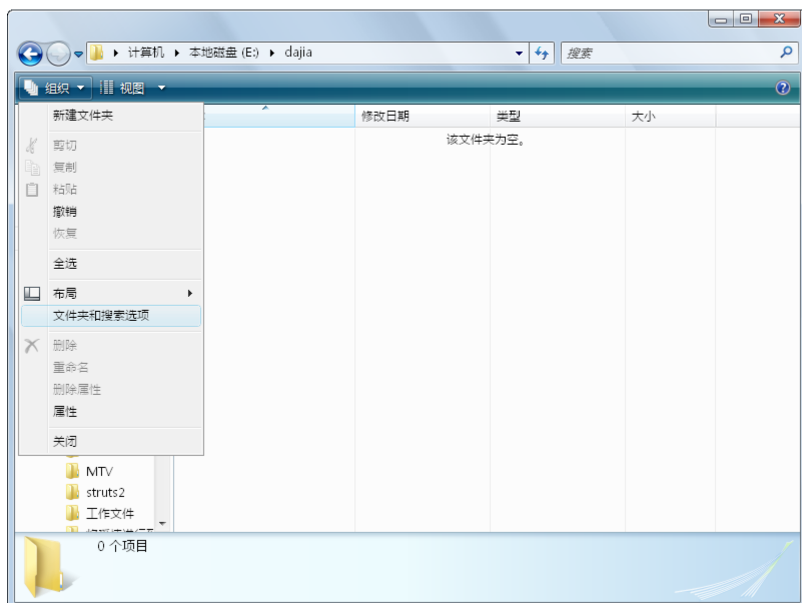


图 1-17　选择命令

2）单击“查看”选项卡，在高级设置列表中取消选择 ☑ 隐藏已知文件类型的扩展名 复选框，之后单击 确定 ，如图 1-18 所示。

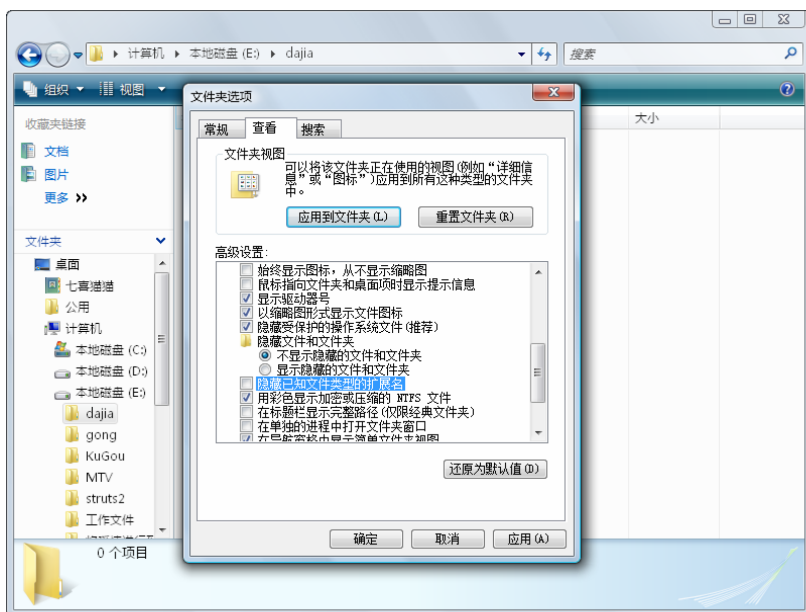


图 1-18　取消选择扩展名

3）新建一个文本文件并用记事本打开，输入下面的程序：

```
class LoveJava{
    /**
    * 这是一个 main 方法
    */
   public static void main(String [] args){
       /* 输出此消息 */
       System.out.println("Java，我喜欢你！");
   }
}
```

4）输入完成后，选择“文件/保存”命令，如图 1-19 所示。

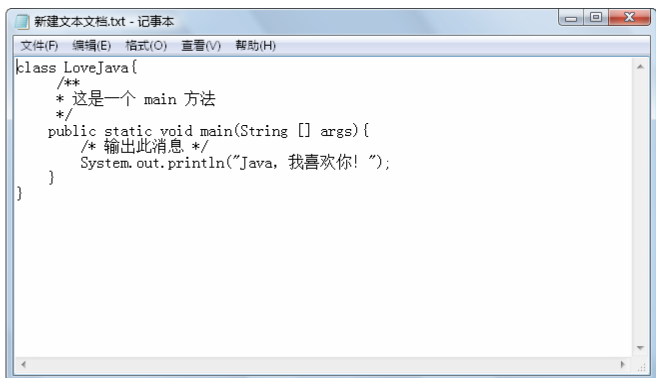


图 1-19　编写的第一个 Java 程序

5）将该文件重命名为“LoveJava.java”，并在命名完成时弹出的“重命名”窗口中单击 是(Y) ，如图 1-20 所示。

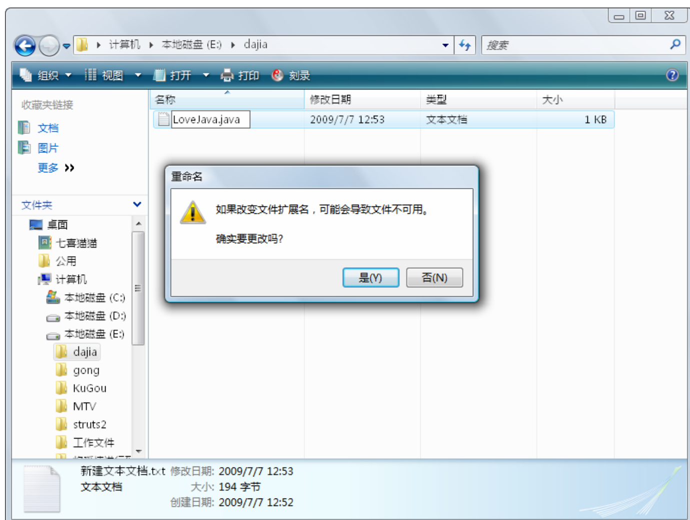


图 1-20　重命名

提示：编写 Java 程序时要特别注意，保证 class 后面的名称和 Java 源文件名对应。

1.4.2 编译和运行 Java 文件

编译和运行 Java 文件十分简单，用户只需在 DOS 环境下进入编写源文件的地方，然后执行两个命令即可，具体操作如下：

1）单击“开始”菜单，选择“运行”命令，在文本框中输入“CMD”并按〈Enter〉键进入 DOS 环境，如图 1-21 所示。



图 1-21 进入 DOS 环境下

2）本地 Java 文件保存在 E 盘的“dajia”文件夹下，用户需要使用 DOS 命令打开这个文件夹，如图 1-22 所示。



图 1-22 使用命令打开文件夹

3）然后输入“javac”和“java”命令对源文件进行编译和运行，操作后如图 1-23 所示。

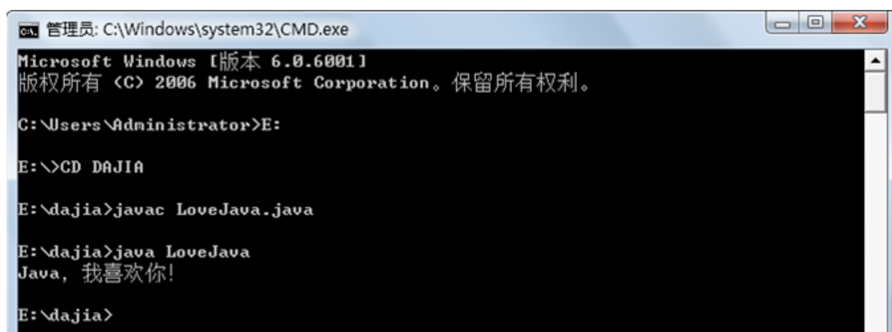


图 1-23 运行和编译 Java 文件

提示：在上面的操作中，使用 DOS 命令打开文件夹的过程十分烦琐，用户可以通过将快捷方式复制到需要打开的文件夹的方法简化操作，具体操作是在 C 盘中搜索 Windows PowerShell 程序，如图 1-24 所示。

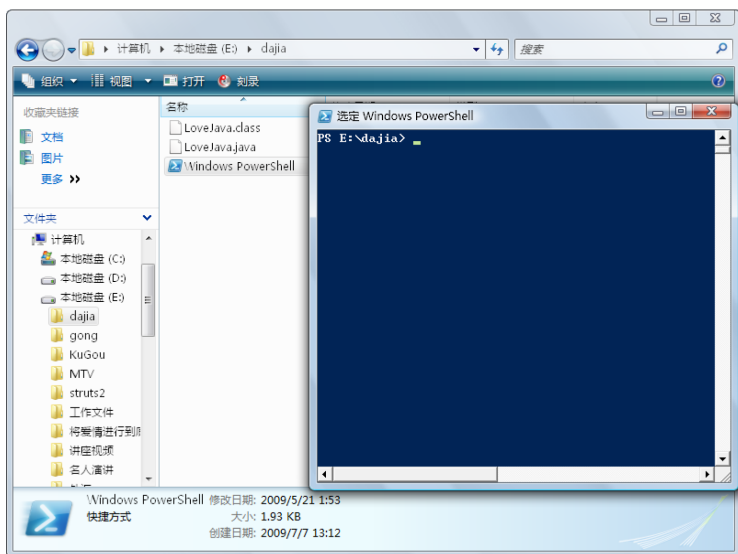


图 1-24　搜索出的快捷方式

1.5　疑难问题解析

本章详细介绍了 Java 的起源、开发环境的基本知识。本节将对本章中较难理解的问题进行着重讲解。

读者疑问：编译和运行 Java 程序的步骤很烦琐，有没有工具可以简化这个过程？开发一个工程，那么多程序，要是都用这个原始的方法一个一个编译运行，效率肯定很低。

解答：的确，如果开发 Java 工程用这么原始的方法编译和执行程序实在是太慢了，也没有效率。Java 是一门优秀的语言，它当然有开发工具，其中 Eclipse 是最有利的利器，它在开发 Java 各个方面，表现得十分出色，这章讲解 Java 的编译和执行，是让读者了解 Java 程序是怎么运行的，在后面的章节中，将会讲解工具的使用。

读者疑问：在编译后，发现多了一个文件，名称和 Java 文件相同，只是扩展名变成了 class，这是怎么一回事？

解答：这是正常的，当用户正确输入 javac 命令时，它就会多出一个 class 的文件，当输入 Java 命令的时候，实际上执行的就是 class 这个文件，这个时候，将扩展名为“.java”的文件删除，执行 java 即可，用户发现这个问题，需要明白，javac 命令编译 Java 源程序，编译后多了一个 class 文件，使用 java 命令执行程序，执行的是 class 文件，而不是 java 源文件。



职场点拨——谈 Java 的重要性

目前，中国对软件人才的需求已达 50 万，并且仍在逐年增长。在未来 5 年时间里，中

国软件人才的需求将远大于供给。最新的统计表明，2011 年我国软件人才的缺口已达 55 万，其中 Java 人才最为缺乏。据悉，在所有软件开发类人才的需求中，对 Java 工程师的需求达到全部需求量的 60%~70%。

在过去两年里，在编程语言使用率排行榜中 Java 一直位居第一，表 1-1 是过去两年的编程语言使用率排行。

表 1-1　2010～2011 年编程语言使用率统计表

2011 年排名	2010 年排名	语　　言	2011 年占有率(%)	和 2010 年相比(%)
1	2	Java	18.058	+2.59
2	1	C	17.051	-1.29
3	3	C++	9.707	-1.03
4	4	PHP	9.662	-0.23
5	5	Visual Basic	6.392	-2.70
6	6	C#	4.435	+0.38

（1）Java 的主要应用领域

Java 的功能十分强大，在服务器领域、移动设备、桌面应用和 Web 领域都占据了重要的地位。

- 桌面应用：Java 和 C++、.NET 一样重要，影响着桌面程序的发展。
- Web 领域：Java Web 有着巨大的优势，无论是开发工具还是开发框架都是开源的，并且安全性高。
- 服务器领域：Java 在服务器编程方面很强悍，拥有很多其他语言所没有的优势。
- 移动设备：Java 在手机领域的应用比较广泛，手机 Java 游戏随处可见，当前异常火爆的 Android 应用开发就使用了 Java 语言。

（2）就业前景

互联网的语言，必然具有其移动性、安全性和开放性的特点，Java 具备以上特点，独受追捧。在以后相当长的一段时间内，采用 Java 编程的 IT 产品的价值将持续增加，估计到 2015 年将达到 45.53 亿美元，年增长率为 14.9%。截止到 2010 年 5 月，Java 注册开发商超过 1300 万人，对 JRE（Java 运行环境）的下载达 9200 万次。詹姆斯•戈士林博士预计在 3～5 年内 Java 技术开发商将发展到 2000 万，无线 Java 开发商也在迅速攀升。

全球有 25 亿 Java 器件运行着 Java，450 多万 Java 开发者活跃在地球的每个角落，数以千万计的 Web 用户每次上网都亲历 Java 的威力。今天，Java 运行在 7.08 亿手机、10 亿智能卡和 7 亿 PC 上，并为 28 款可兼容的应用服务器提供了功能强大的平台。如此多的应用，彻底改变了人们的生活。越来越多的企业，因为使用了 Java 而提高了生产效率。越来越多的用户，因为 Java 而降低了成本，享受了生活。

第 2 章　Java 开发工具介绍

在上一章编写 Java 程序的过程中不难发现，依靠原始方法编写程序不但操作复杂，效率也很低，这就需要借助开发工具来提高开发效率。本章将介绍几种主流的开发工具。本章主要内容如下：

- ❑ Java 工具简介。
- ❑ Eclipse 的获得与安装。
- ❑ NetBeans 的使用。
- ❑ 职场点拨——学习 Java 的正确态度。

2008 年 X 月 XX 日，天气阴

学习 Java 一段时间了，我感觉很复杂，要想完全掌握需要付出很大精力。闲来无事的时候我经常想要是有一个快速学习 Java 的方法就好了……



一问一答

小菜：“有学习 Java 的捷径吗？”

Wisdom：“学习编程没有捷径，欲速则不达，你应该端正学习态度。在本章末我会重点介绍学习 Java 的方法。古人云“工欲善其事，必先利其器”，作为初学者的你，选择合适的开发工具十分重要。借助优秀的开发工具编写程序能起到事半功倍的效果。”

小菜：“我应该用什么工具？”

Wisdom：“开发 Java 的 IDE 工具很多，其中人们使用最多的是 Eclipse，其特点是完全免费，并且不需要安装。”

2.1　认识 Java 的开发工具

开发 Java 的 IDE 工具很多，其中最为著名的是 Eclipse，其特点是绿色、不需要安装，且用户可以根据自己的需要，寻找或者制作插件，配置在 Eclipse 中，以强化它的功能。如图 2−1 所示为 Eclipse 工具的界面。



图 2-1 Eclipse 的开发界面

2.2 Java 开发工具简介

Java 是一门优秀的跨平台的程序语言，很多公司专为其研制了开发工具，其中不乏一些优秀的作品，而最为出众的是 Eclipse、NetBean 和 JBuilder 这款个集成开发工具，下面将对它们分别进行详细的介绍。

2.2.1 Eclipse 简介

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。就其本身而言，它只是一个通过插件组件构建开发环境的框架，但众多的插件支持使 Eclipse 拥有更多相对于其他固定 IDE 工具更为突出的功能。Eclipse 附带了一个标准的插件集，包括 Java 开发工具（Java Development Tools，JDT）。

虽然大多数用户乐于将 Eclipse 当做 Java IDE 工具来使用，但 Eclipse 的功能不仅限于此。Eclipse 还包括了插件开发环境（Plug-in Development Environment，PDE），这个组件主要针对希望扩展 Eclipse 的软件开发人员，因为它允许他们构建与 Eclipse 环境无缝集成的工具。由于 Eclipse 中的每一个元素都是插件，因此对于给 Eclipse 提供插件，以及给用户提供一致和统一的集成开发环境而言，所有工具开发人员都具有同等的发挥舞台。

Eclipse 是著名的跨平台的自由集成开发环境（IDE），最初主要用于 Java 语言开发，如今亦有人通过插件使其作为其他计算机语言比如 C++和 Python 的开发工具。还有许多软件开发商以 Eclipse 为框架开发自己的 IDE，如图 2-2 所示为 Eclipse 启动界面。

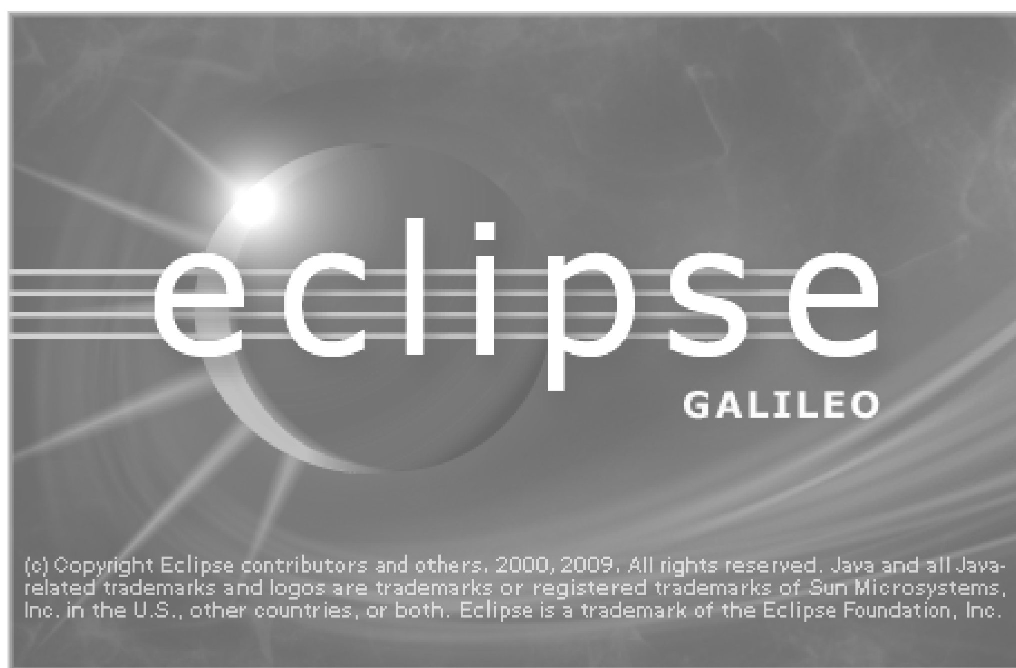


图 2−2　Eclipse 启动界面

2.2.2　Netbeans 简介

NetBeans 是由 Sun 公司研制的开放源码的软件开发工具，是一个开放框架，一个可扩展的开发平台，可以用于 Java、C 和 C++等的开发。其本身是一个开发平台，和 Eclipse 一样，NetBeans 也可以通过扩展插件来扩展功能。

NetBeans 是 Sun 公司在 2000 年创立的，它是开源运动以及开发人员和客户社区的家园，旨在构建世界级的 Java IDE。目前 NetBeans 可以在 Solaris、Windows、Linux 和 Macintosh OS X 平台上进行开发，并在 SPL（Sun 公用许可）范围内使用。http://www.netbeans.org 已经获得业界广泛认可，并支持 NetBeans 扩展 IDE 模块目录中大约 100 多个模块。

NetBeans 是一个全功能的开放源码 Java IDE，可以帮助开发人员编写、编译、调试和部署 Java 应用，并将版本控制和 XML 编辑融入其众多功能之中。NetBeans 可支持 Java 2 平台标准版（J2SE）应用的创建、采用 JSP 和 Servlet 的 2 层 Web 应用的创建，以及用于 2 层 Web 应用的 API 及软件的核心组的创建。此外，NetBeans 最新光盘还预装了两个 Web 服务器，即 Tomcat 和 GlassFish，从而免除了烦琐的配置和安装过程。所有这些都为 Java 开发人员创造了一个可扩展的开源多平台的 Java IDE。

NetBeans 只有基本的 Java 开发功能，如果需要更多的开发功能，可通过添加平台提供的开发组件来完成，如建立桌面应用、企业级应用、Web 开发和 Java 移动应用程序开发、C/C++等。NetBeans 支持多种操作系统平台，包括 Windows、Linux、Mac OS 和 Solaris 等操作系统，如图 2−3 所示为运行中的 Netbeans。

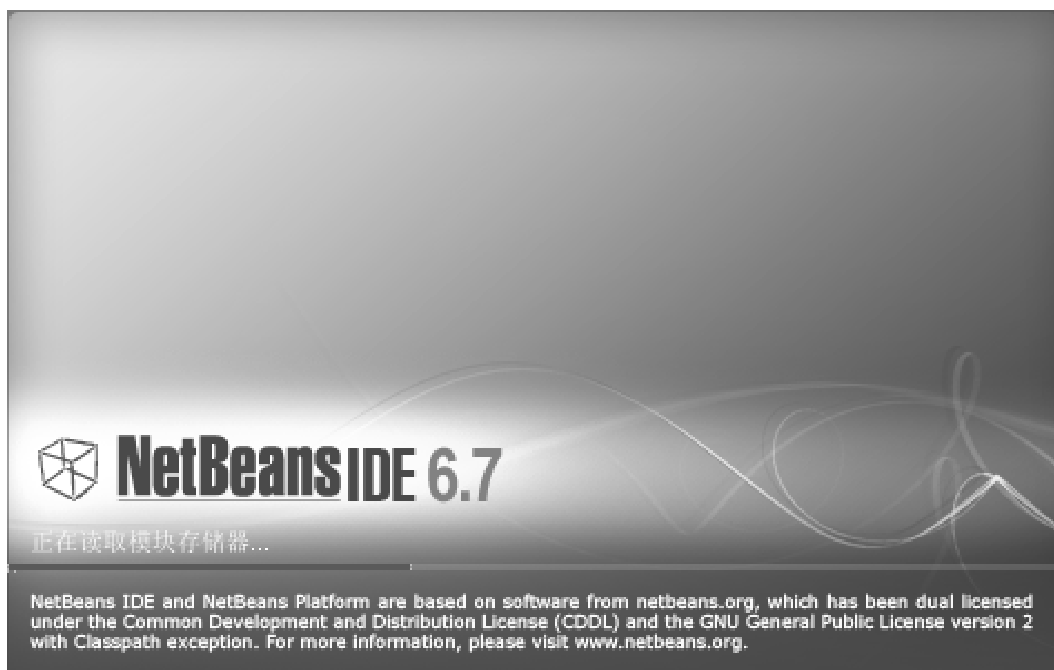


图 2-3　NetBeans 的欢迎界面

2.2.3　JBuilder 简介

JBuilder 是 Borland 公司开发的针对 Java 的开发工具，使用 JBuilder 可以快速、有效地开发各类 Java 应用程序，它使用的 JDK 与 Sun 公司标准的 JDK 不同，它经过了较多的修改，以便开发人员能够像开发 Delphi 应用那样开发 Java 应用。

JBuilder 的核心有一部分采用了 VCL 技术，利用它编写的代码即使是初学者也可以读懂。JBuilder 的另外一个好处就是可以通过互联网，将分布在世界各地的开发人员联合起来开发项目。总的说来，它具有下面的优点：

- JBuilder 支持最新的 Java 技术，包括 Applet、JSP/Servlet、JavaBean 以及 EJB（Enterprise JavaBean）的应用。
- 用户可以自动地生成基于后端数据库表的 EJB Java 类，JBuilder 同时还简化了 EJB 的自动部署功能。此外它还支持 CORBA，相应的向导程序有助于用户全面地管理 IDL（分布应用程序所必需的接口定义语言）和控制远程对象。
- JBuilder 支持各种应用服务器，JBuilder 与 Inprise Application Server 紧密集成，同时支持 WebLogic Server，支持 EJB 1.1 和 EJB 2.0，可以快速开发 J2EE 的电子商务应用。
- JBuilder 能用 Servlet 和 JSP 开发和调试动态 Web 应用。
- 利用 JBuilder 可创建纯 Java2（没有专有代码和标记）应用。JBuilder 是用纯 Java 语言编写的，其代码不含任何专属代码和标记，它支持最新的 Java 标准。
- JBuilder 拥有专业化的图形调试界面，支持远程调试和多线程调试，调试器支持各种 JDK 版本，包括 J2ME/J2SE/J2EE。JBuilder 环境开发程序方便，是纯 Java 开发环境，适合企业的 J2EE 开发；缺点是往往一开始人们难于把握整个程序各部分之间的

关系，对机器的硬件要求较高，运行速度较慢。

2006 年 JBuilder 被划归到 Borland 全资子公司 CodeGear，2008 年 5 月 7 日 Borland 以 2300 万美元卖掉 CodeGear 部门，买家是 Embarcadero 公司。

2.3　Eclipse 的获得与安装

Eclipse 是一个开放源代码的软件开发项目，专注于为高度集成的开发工具提供一个全功能的、具有商业品质的工业平台。它主要由 Eclipse 项目、Eclipse 工具项目和 Eclipse 技术项目三个项目组成，具体包括四个部分：Eclipse Platform、JDT、CDT 和 PDE。JDT 支持 Java 开发、CDT 支持 C 开发、PDE 支持插件开发，而 Eclipse Platform 则是一个开放的可扩展 IDE，它提供了一个通用的开发平台，下面介绍它的获得与安装方法。

2.3.1　获得 Eclipse

Eclipse 是一款免费的开发工具，用户可以去官方网站下载获取，具体操作如下：

1）打开 IE 浏览器，在地址栏中输入网址“http://www.Eclipse.org/”，然后单击“Download Eclipse”超级链接，如图 2-4 所示。

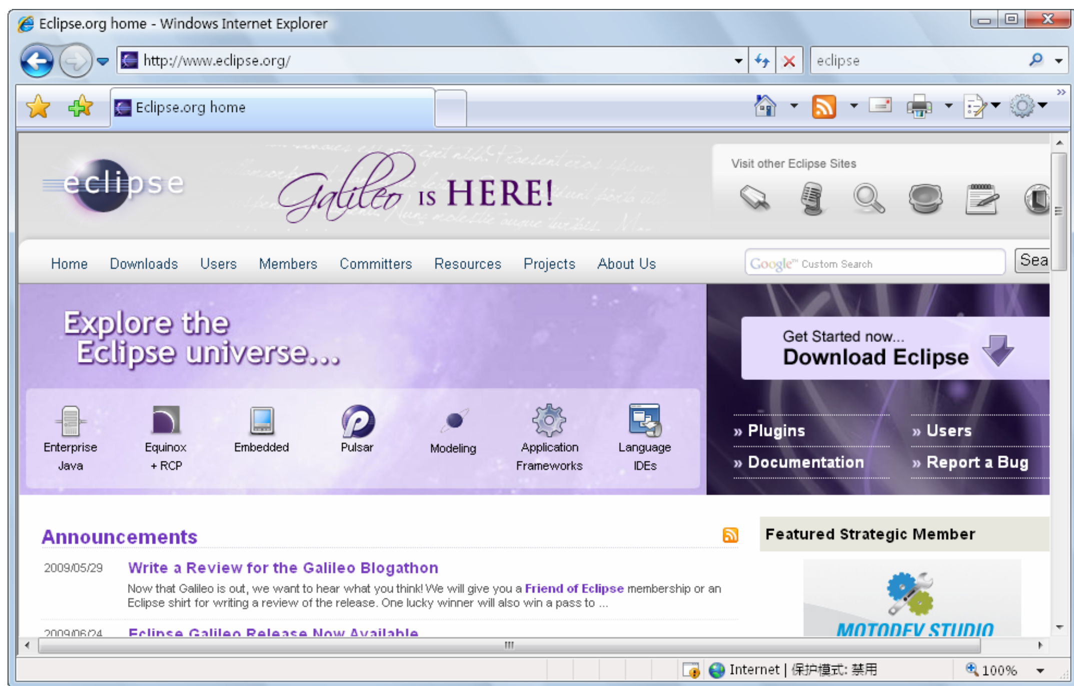


图 2-4　Eclipse 首页

2）进入“Eclipse”下载页面，根据操作系统选择对应的“Eclipse”版本，这里选择“Windows”版本，故单击“Windows”超级链接，如图 2-5 所示。

3）单击“[China] Actuate Shanghai (http)”超级链接，即可下载 Eclipse。这里启动了“迅雷”进行下载，如图 2-6 所示。

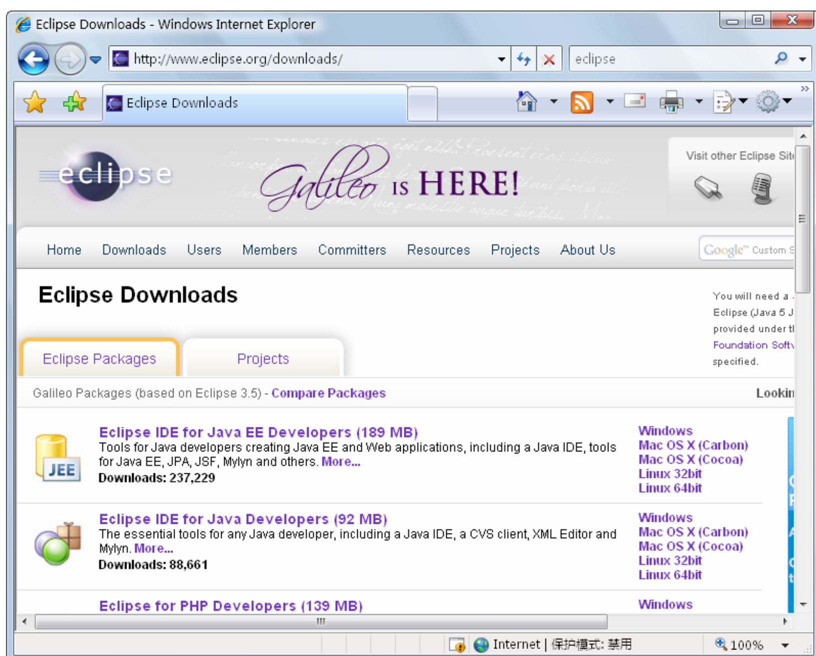


图 2-5　下载页面

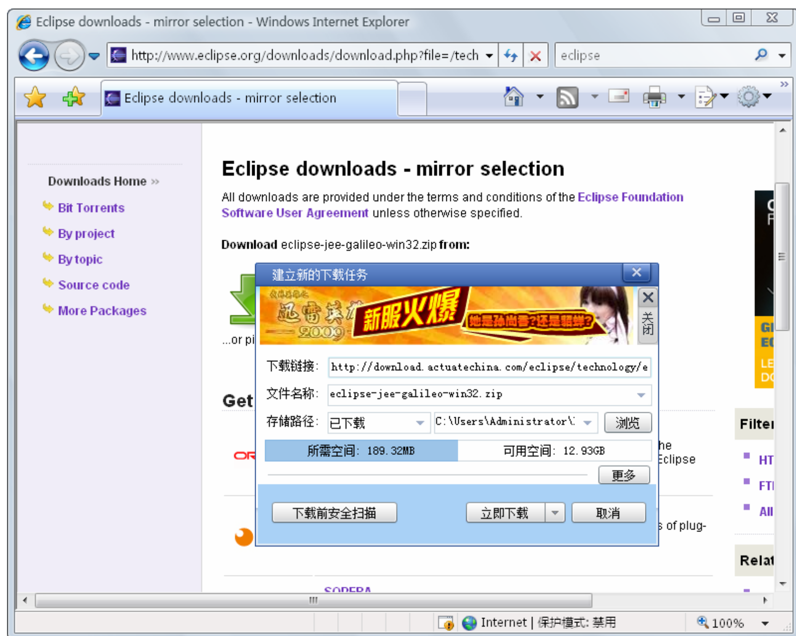


图 2-6　下载 Eclipse

2.3.2　新建一个 Eclipse 项目

下载完成后，将下载的压缩文件解压，然后打开解压出的文件夹，双击 Eclipse.exe 文件，即可启动 Eclipse。对它进行操作，并新建项目，具体操作如下：

1）在任意一个磁盘里新建一个文件夹，这里在 F 盘新建 open 文件夹。然后在“Workspace”文本框中输入所建文件夹的路径，单击“OK”按钮，如图 2-7 所示。

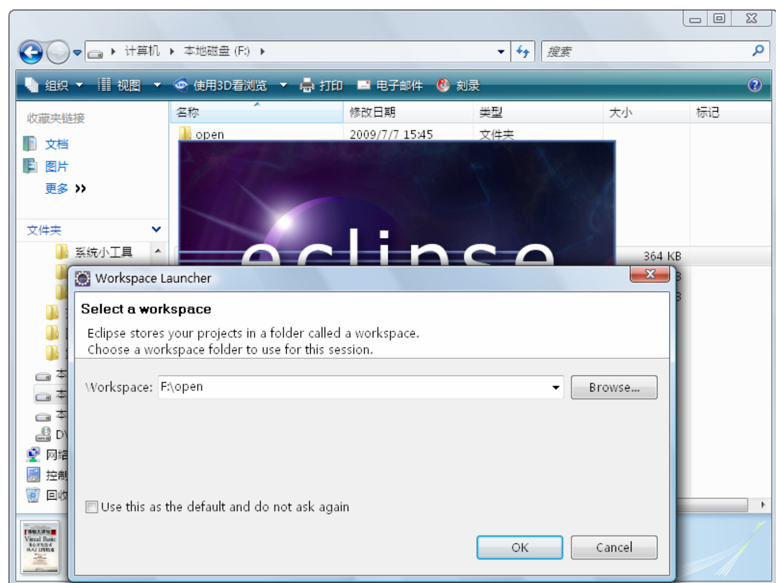


图 2-7　指定项目文件夹

2）选择“File/New/Project”命令新建项目，如图 2-8 所示。

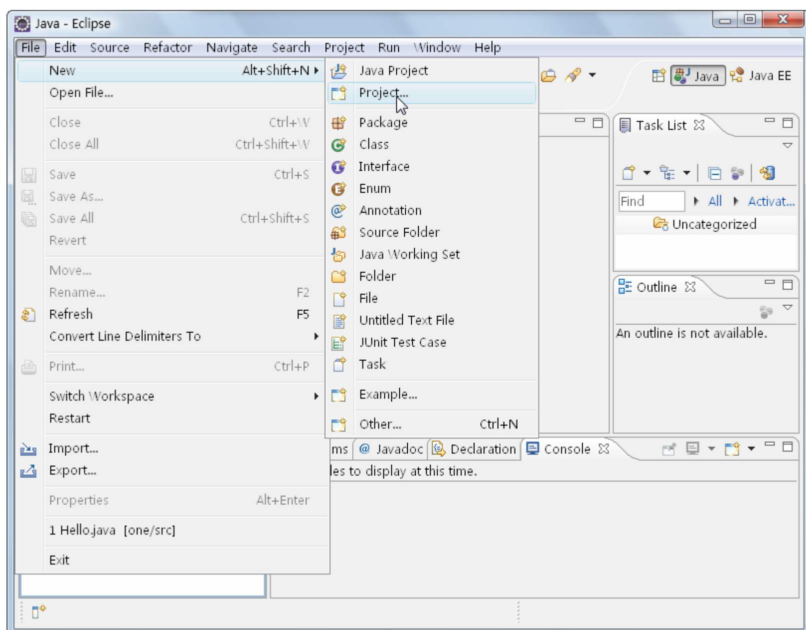


图 2-8　选择命令

3）打开“New Project”窗口，单击“Java”选项，在下拉菜单中打开“Java Project”选项，选中后单击“Next”按钮，如图 2-9 所示。

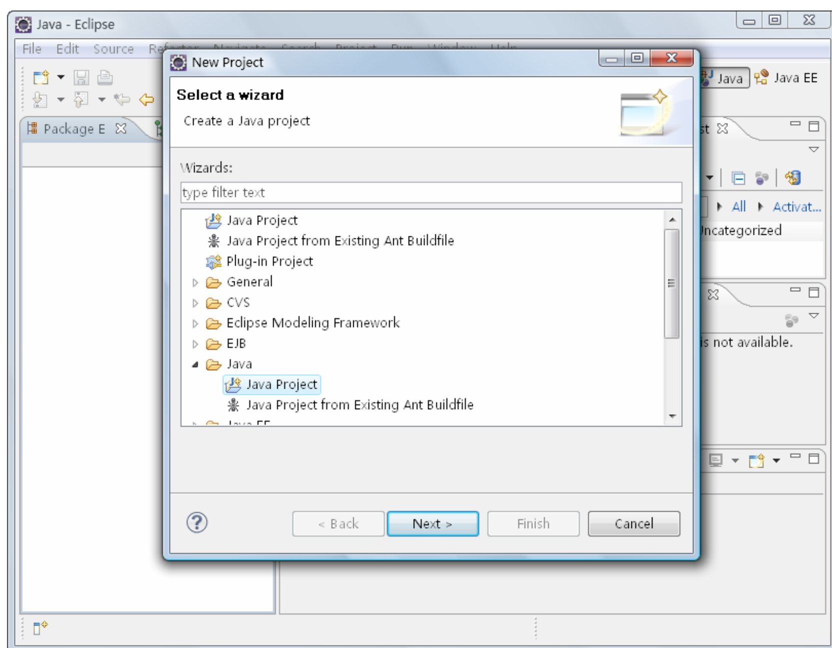


图 2-9 “New Project”窗口

4）在弹出的“Create a Java Project”窗口中的“Project name”文本框中输入项目名称，这里输入“one”，输入完成后，单击“Finish”按钮，如图 2-10 所示。

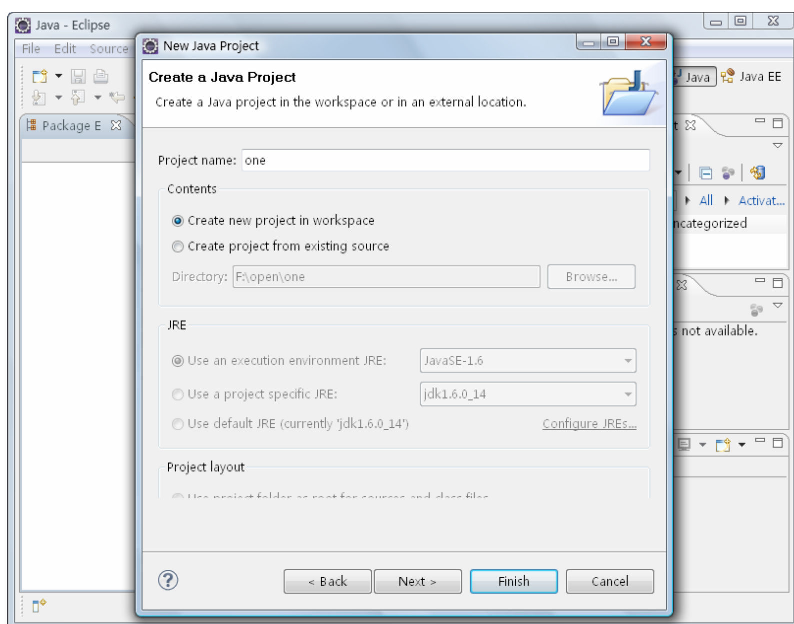


图 2-10 新建项目

5）在 Eclipse 左侧的列表中双击打开“One”项目，选中其中的“str”选项并单击鼠标右键，在弹出的快捷菜单中选择“New/Class”命令，如图 2-11 所示。

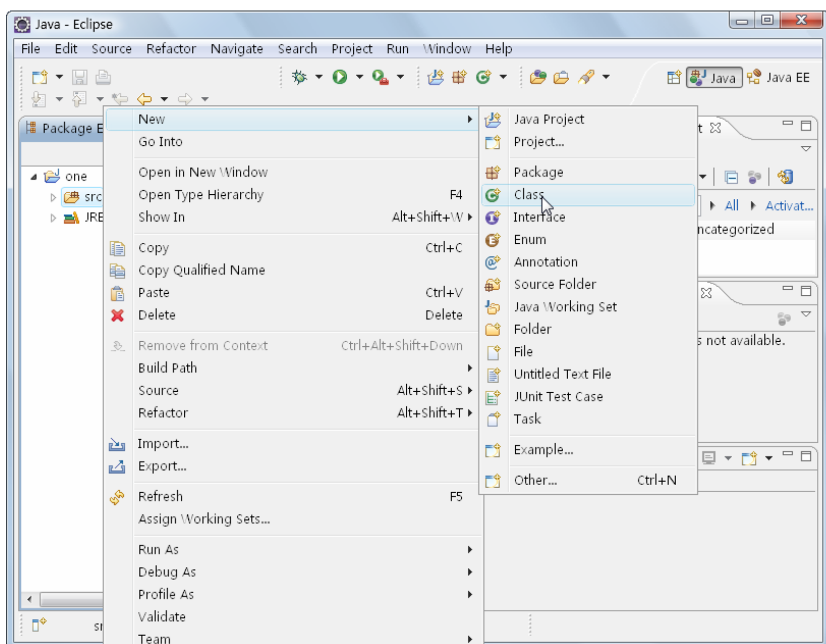


图 2-11　选择命令

6）在弹出的“Java Class”窗口中的“Name”文本框中输入类名，这里输入“Hi”，之后选择☑ public static void main(String[] args) 复选框，单击“Finish”按钮，如图 2-12 所示。

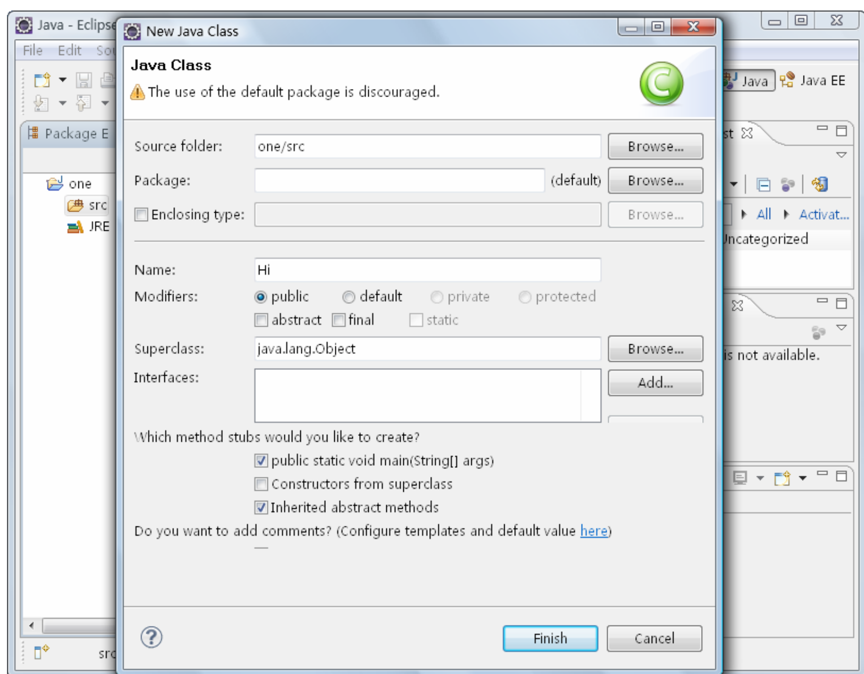


图 2-12　“Java Class”窗口

7）打开“Hi”文件输入代码，输入完成后的效果如图 2-13 所示。

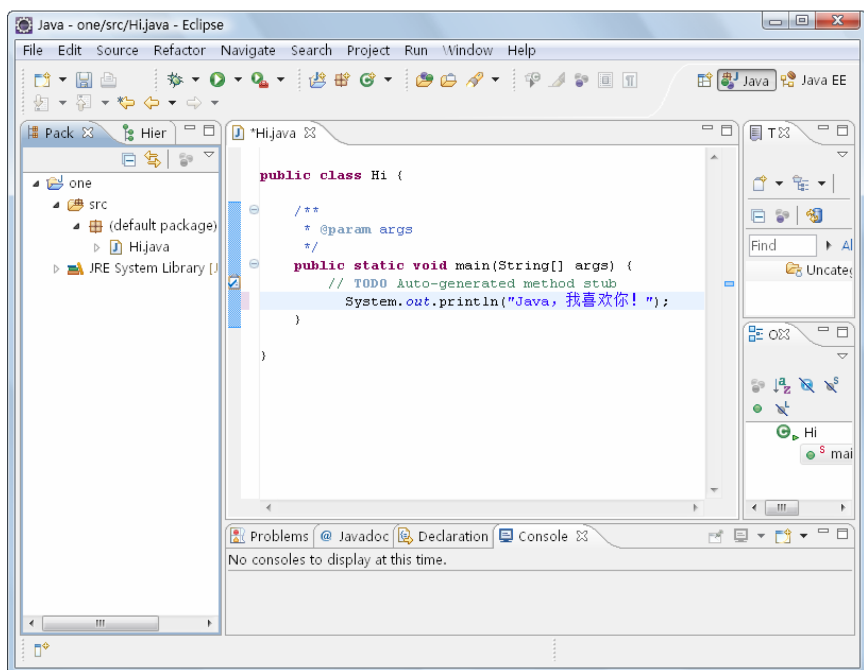


图 2-13　输入代码

8）编译代码后单击按钮，打开“Sava and Launch”窗口，单击“OK”按钮，如图 2-14 所示。

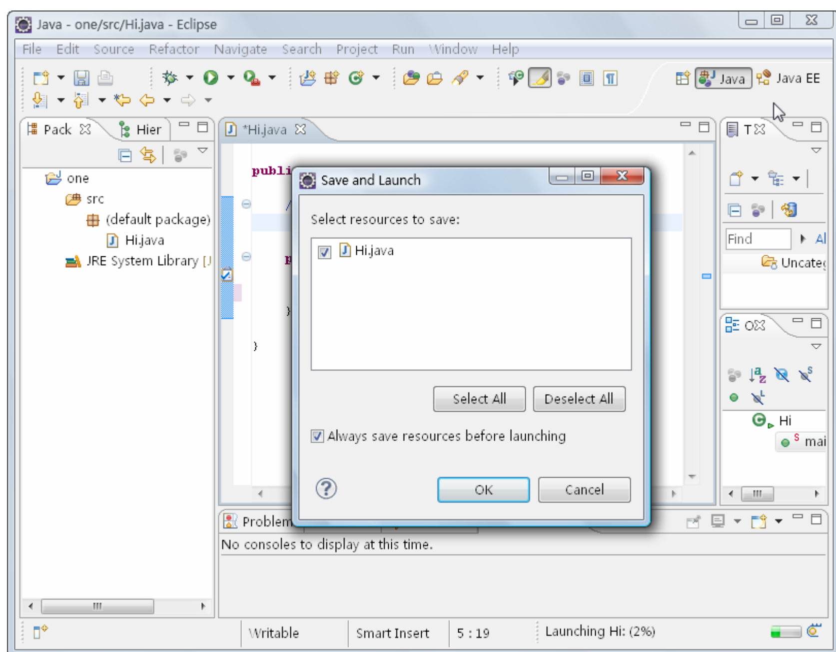


图 2-14　编译 Java

9）单击“OK”按钮即可看到运行后的结果，如图 2-15 所示。

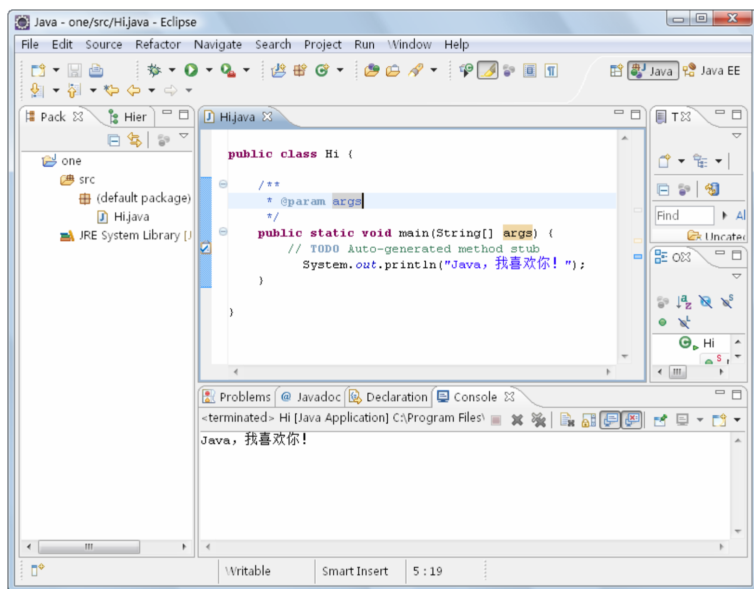


图 2-15　运行后的结果

2.4　NetBeans 的使用

2.4.1　下载 NetBeans

NetBeans 是一款免费的软件，用户可以去官方网站下载，具体操作如下：

1）在 IE 地址栏中输入“http://www.netbeans.org/”，在打开的页面中单击“Download NetBeans IDE”超级链接，如图 2-16 所示。

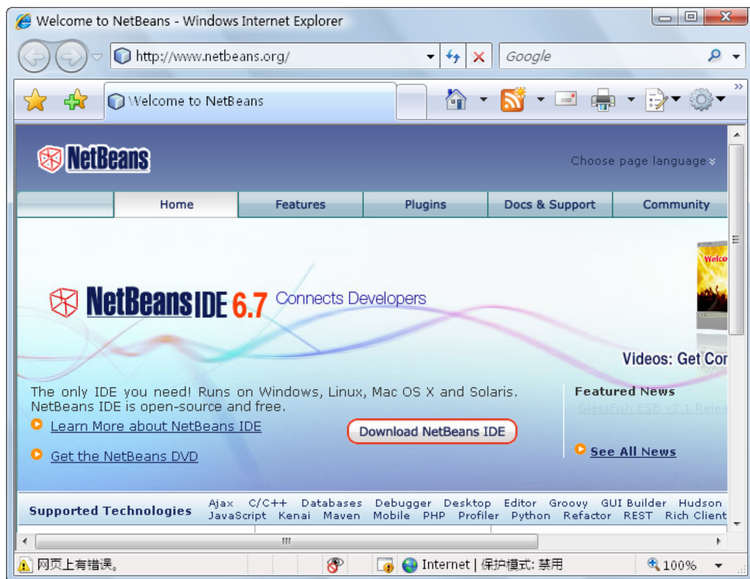


图 2-16　单击超级链接

2）在打开的页面中，用户可以根据需要进行下载，这里单击“All”下的“下载”按钮下载，如图 2–17 所示。

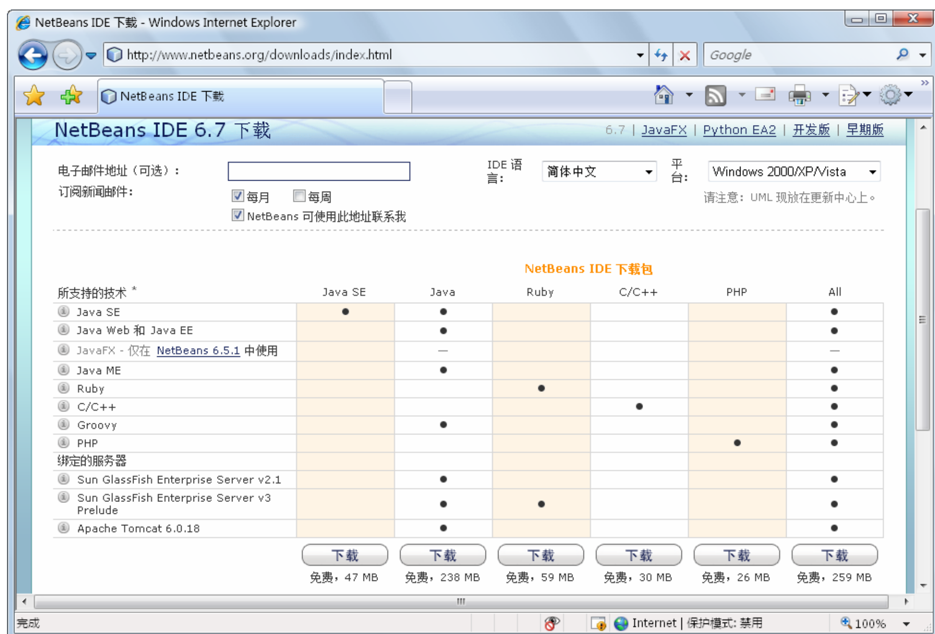


图 2–17　下载“NetBeans IDE”

3）单击“请点击这里下载”超级链接进行下载，这里使用“迅雷”下载，如图 2–18 所示。



图 2–18　启动迅雷进行下载

2.4.2　安装 NetBeans

下载完成后，即可安装 NetBeans，安装过程十分简单，具体操作如下：

1）双击 NetBeans.exe 文件，即可打开“NetBeans”安装程序，单击“定制”按钮，如图 2-19 所示。

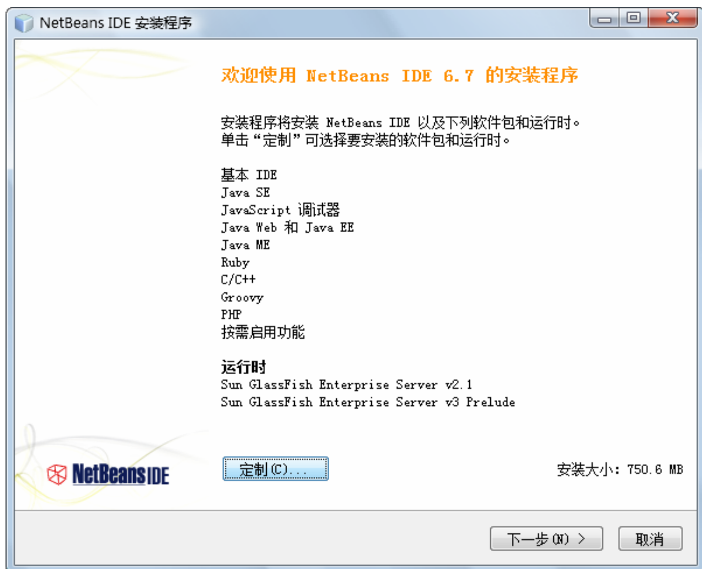


图 2-19　欢迎界面

2）在“定制安装”窗口中的列表里选择需要的组件，之后单击“确定”按钮，如图 2-20 所示。



图 2-20　选择组件

3）返回到欢迎界面，单击“下一步”按钮，在打开的“许可证协议”窗口中仔细阅读协议后，选中☑我接受许可证协议中的条款(A)复选框，并单击“下一步”按钮，如图 2-21 所示。

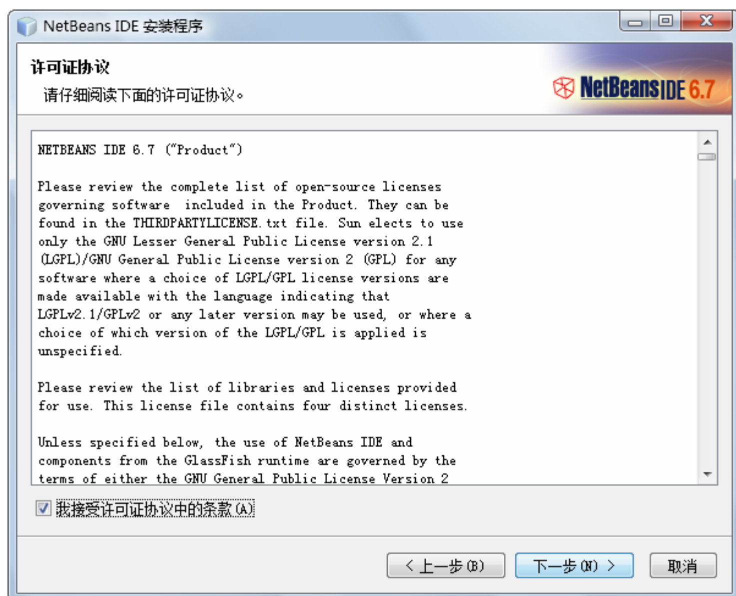


图 2-21 “许可证协议”窗口

4）在打开的“NetBeans IDE 6.7 安装”窗口中为 NetBeans IDE 和用于 NetBeans IDE 的 JDK 指定安装路径，单击“下一步”按钮，如图 2-22 所示。

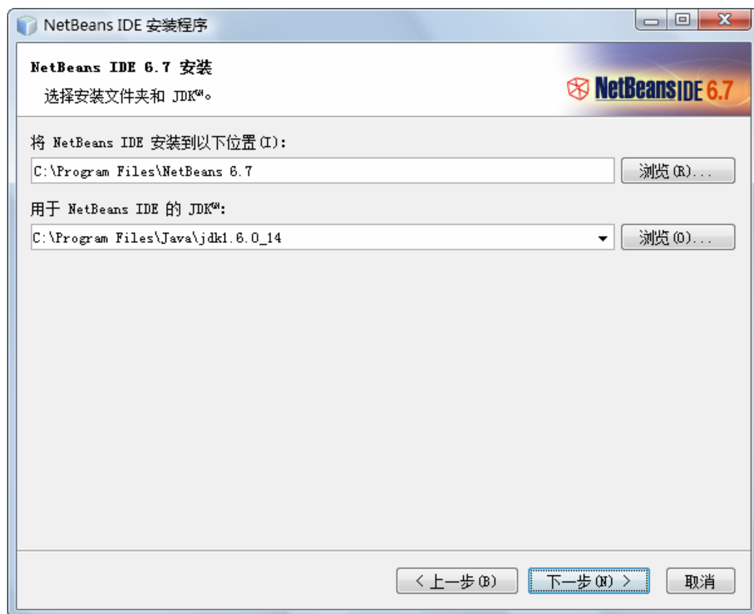


图 2-22 安装“NetBeans IDE”窗口

5）在打开的“Apache Tomcat 6.0.18 安装”窗口中指定 Apache Tomcat 的安装路径，单

击“下一步”按钮，如图 2-23 所示。

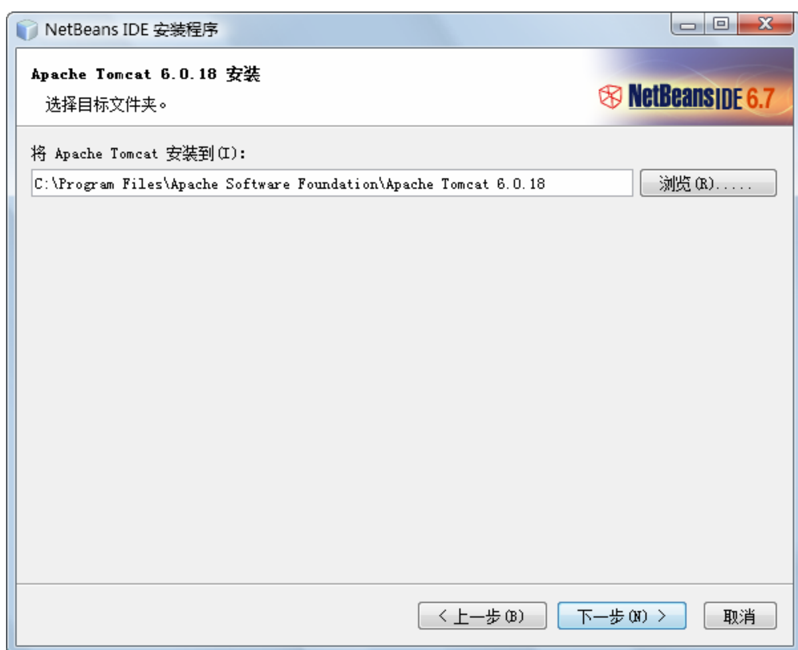


图 2-23　指定 Tomcat 的位置

6）在打开的“摘要”窗口中单击“安装”按钮开始安装，如图 2-24 所示。

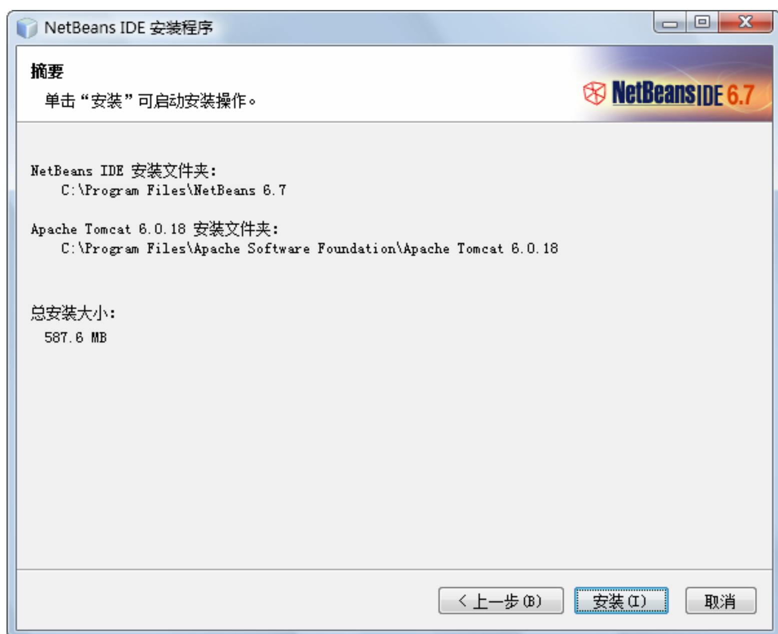


图 2-24　安装文件

7）安装需要一定的时间，用户需耐心等待，如图 2-25 所示。

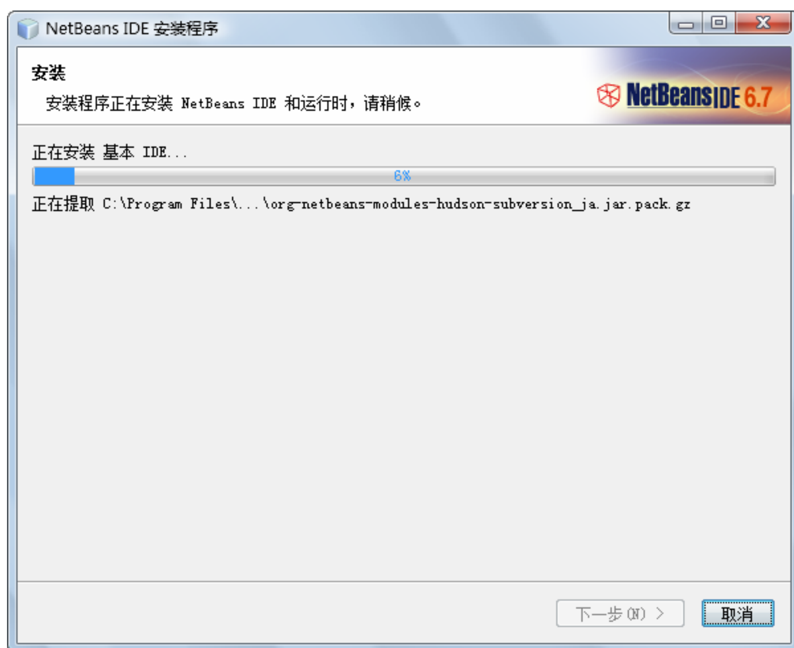


图 2-25　安装 Netbeans

8）安装度读条读取完毕后，单击“下一步”按钮，打开“安装完成”窗口，选择 ☑ 通过提供匿名使用数据，帮助改进 NetBeans 项目 复选框和 ☑ 安装 NetBeans IDE 后对其进行注册 复选框，然后单击“完成”按钮完成安装，如图 2-26 所示。

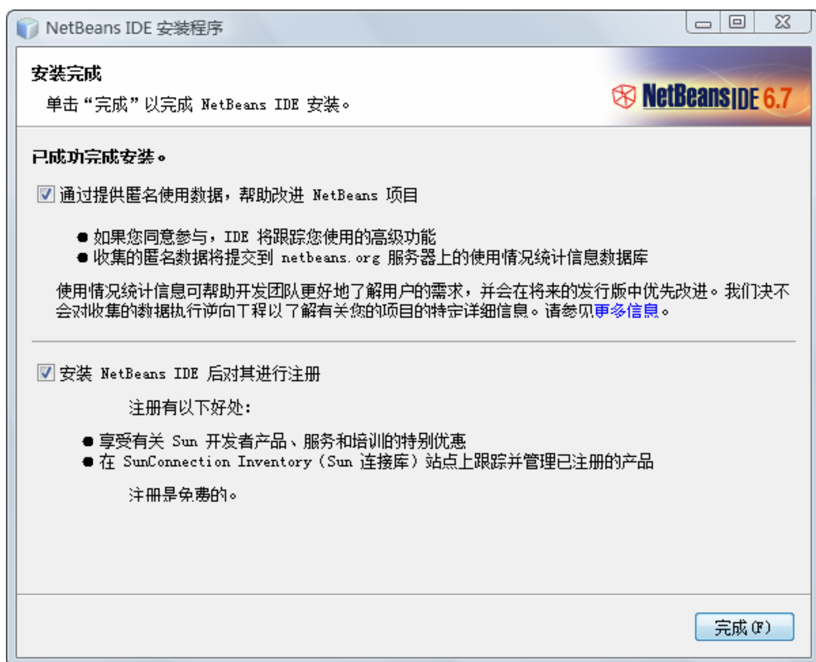


图 2-26　完成安装

2.4.3　使用 NetBeans 新建项目

NetBeans 是 Sun 公司开发的 Java 工具软件，使用 NetBeans 新建项目的操作方法十分简单，具体操作如下：

1）双击 NetBeans 图标，启动 NetBeans 软件，如图 2-27 所示。



图 2-27　NetBeans IDE 软件

2）选择“文件/新建项目”命令，或按〈Ctrl+Shift+N〉快捷键即可新建项目，如图 2-28 所示。

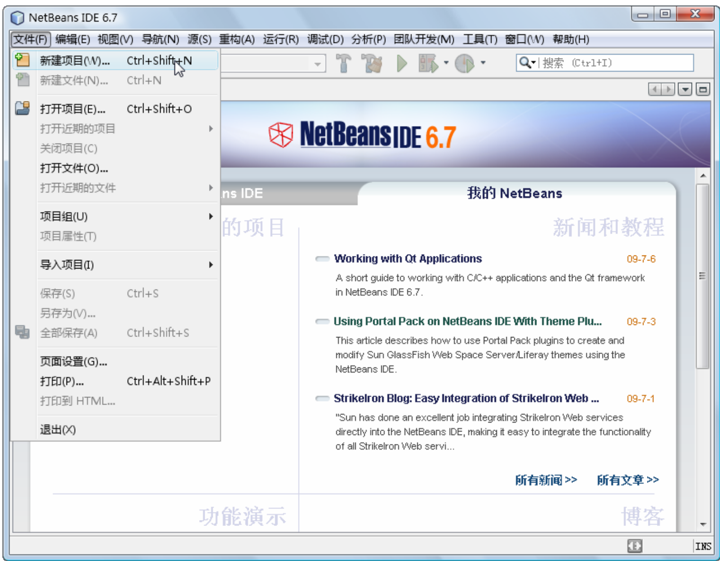


图 2-28　新建项目

3）打开“新建项目”窗口，在左侧“类别”列表中选择需要的类，这里选择 Java，然后在“项目”列表中选择项目的类别，这里选择 Java 应用程序，如图 2–29 所示。

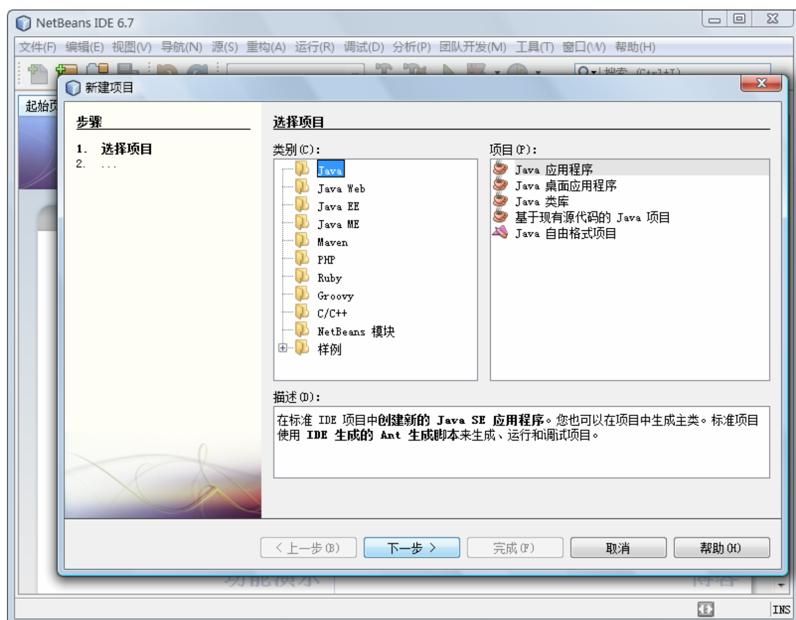


图 2–29　选择类和项目类别

4）选择后单击“下一步”按钮，在打开的“新建 Java 应用程序”窗口中设置项目的名称、项目位置以及创建主类的名称，如图 2–30 所示。

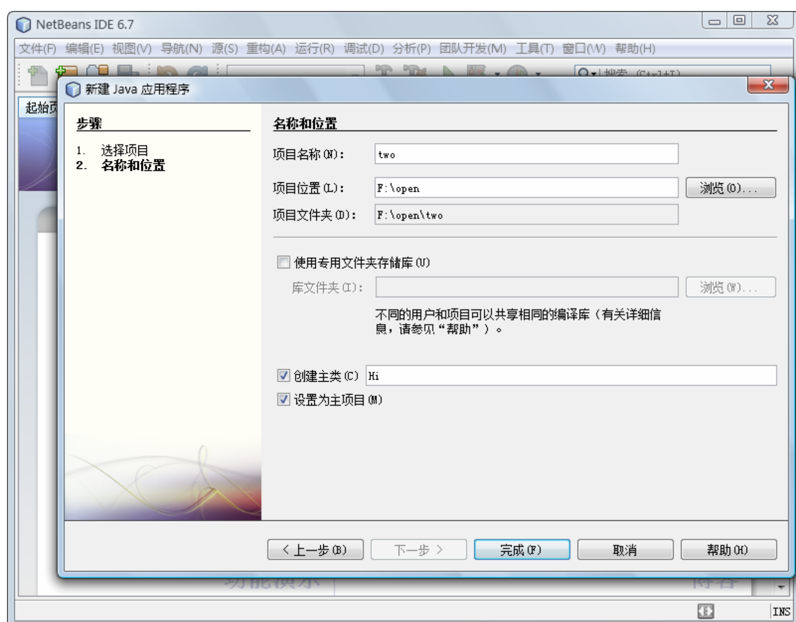


图 2–30　创建项目

5）进入项目的编写窗口，打开新建的类，编写程序。然后单击按钮，进行调试运行，得到的结果如图 2−31 所示。

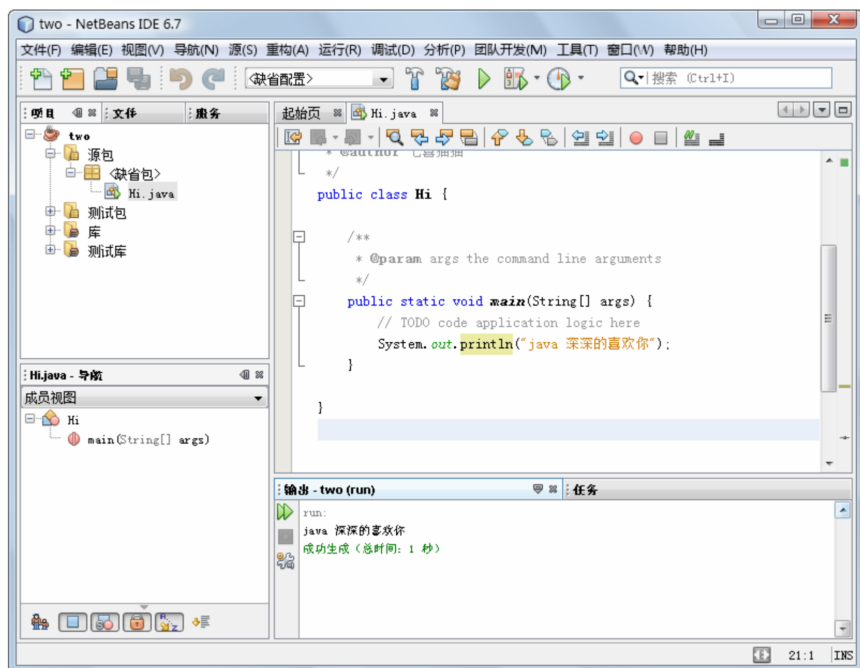


图 2−31　NetBeans 调试的结果

2.5　疑难问题解析

本章详细介绍了开发工具 Elipse、NetBeans 的基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：Eclipse 和 NetBeans，它们都是开发 Java 工具，谁的功能强大一些？

解答：NetBeans 是 Java 官方提供的开发工具，Eclipse 是第三方公司提供的开发工具，两者针对不同的开发领域，其特点也各有千秋。国内外，绝大多数 Java 开发者还是选择使用 Eclipse 开发。

读者疑问：在 Java 论坛里发现有人讨论 JBuilder 的开发工具，它是什么开发工具？与 Elipse、NetBeans 有什么区别？

解答：JBuilder 是较早的 Java 开发工具，是收费软件，相比 Elipse、NetBeans 而言功能比较简单，它的市场几乎被 Elipse 完全抢占。目前已经很少有开发者使用 JBuilder 了。



职场点拨——学习 Java 的正确态度

软件开发是一件苦中求乐的事情，而学习软件开发更是如此。经验丰富的程序员传授给

初学者的经验是——打好基础、多实践。在此笔者提出如下 3 条建议，目的是希望大家提高学习 Java 的效率。

（1）培养兴趣

无论做什么事情，只要有了兴趣，人们就喜欢花费时间去做它。只要喜欢感受那调试成功的喜悦，就说明已经对编程产生了兴趣。作为一名程序员，要懂得分享。建议大家在闲暇时多去一些程序员论坛转转，分享自己的得与失，体会别人的苦与乐。在论坛上有很多乐于助人的高手，他们能够帮助“菜鸟”解决很多技术问题。

（2）脚踏实地

都说“欲速则不达”，学习编程切忌浮躁的心态。有许多初学者刚刚学会了基本语法知识，调试成功了几段代码，就迫不及待地大声宣布“我精通 Java 了”。但是真正在工作岗位上面对一个个大型项目时，才发现自己学到的只是九牛一毛。

（3）多实践

软件开发很强调实践能力，很多前辈认为学习编程的秘诀就是：编程、编程、再编程，练习，练习，再练习！笔者对此深表赞同。学编程不仅要多实践，而且要快速实践。读者在看书的时候，不要等到完全理解了才动手，而是应该在看书的同时输入代码，程序运行的各种情况可以让读者更快更牢固地掌握知识点。

第 3 章　Java 数据

数据对每一种程序设计语言来说都是十分重要的，优秀的程序设计语言对数据的处理都有其独到之处。本章将讲解 Java 数据的基本知识、变量、常量和数据类型等内容。本章主要内容如下：

- 量。
- 数据类型。
- 标识符和关键字。
- 职场点拨——不同客户，不同的处理方式。

算上今天，小菜已在产品部客户服务这个职位上奋斗了整整一个月。

2010 年 X 月 X 日，多云

明天是周三，听说将有客户来访，这是参加工作以来第一次见客户，心里难免有些紧张和不知所措，甚至连穿什么衣服都难以抉择。



一问一答

Wisdom：“看你无精打采的，在为什么事情发愁呢？”

小菜：“明天是我第一次独自接见客户，心里没底……”

Wisdom：“其实客户很好相处，只要你了解了对方是什么类型的客户，来访的目的是什么，你就能知道客户想要什么，想听什么。相信在本章最后的‘不同客户，不同的处理方式’会对你有所帮助。”

3.1　量

量是用来传递数据的介质，它的作用十分重要。程序中的量既可以是变化的，也可以是固定的，依其本身的性质可分为变量和常量，下面将进行详细讲解。

3.1.1　常量

永远不变的量就是常量，其值不能改变。常量可以是不随时间变化的某些量和信息，也可以是表示某一数值的字符或字符串。在 Java 程序中，常量名常用大写字母表示，而 value

则是该数据合法的值，如下面的代码，就是常量的格式：

```
final double PI=value;
```

代码 1：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/Test.java”。首先定义 4 个类（类的具体知识将在本书后面内容讲解），具体代码如下：

```
class X                                    //定义类 x
{
   public static String strX="hello";
}
class Y                                    //定义类 y
{
   public static String strY="hello";
}
class Z                                    //定义类 z
{
   public static String strZ="hell"+"o";
}
```

然后定义类 Test，在类 Test 中定义两个 String 类型的常量，输出结果。代码如下：

```
public class Test
 {
   public static void main(String[] args)
      {
          System.out.println( X.strX==Y.strY);           //输出结果
          System.out.println( X.strX==Z.strZ);           //输出结果
          String s1="hel";                               //给 s1 赋值
          String s2="lo";                                //给 s2 赋值
          System.out.println( X.strX==(s1+s2));          //输出结果
          System.out.println( X.strX==(s1+s2).intern()); //输出结果
       }
}
```

类 X，类 Y，类 Z 中的三个常量字符串属于不同的对象，用 = = 操作符比较，其结果必然是 false，如图 3-1 所示。

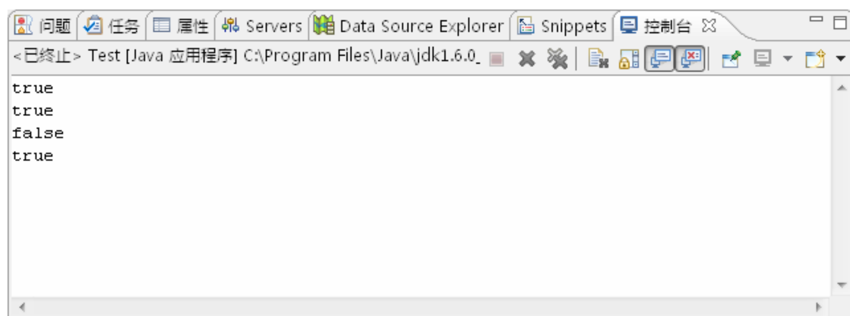


图 3-1　常量

3.1.2　变量

为变而生的量，叫做变量。在此需要声明的是，但凡变量都必须分配一个类型，在任何程序设计中均如此。在程序运行过程中，空间内的值是变化的，这个变化的空间就称为变量。为了方便，人们给这个空间取了个名字，叫“变量名”。因为内存空间内的值本身就是变量值，所以申请的内存空间的变量不一定有值，要想变量有值，就必须要放入值。不过对于申请的变量，每一个数据类型都会有一个默认值，如 int 的数据变量的默认值是“0”，char 的数据变量的默认值是“null”，byte 的数据变量的默认是“0”。

程序运行中，变量的基本格式与常量有所不同，代码如下：

```
typeSpencifier varName=value;
```

参数介绍：

- typeSpencifier 为 Java 中合法的数据类型，这和常量是一样的。
- varName 为变量名，变量和常量的最大区别在于 value，对于变量来说，value 的值是可有可无，而且还可以对其进行动态初始化。

变量又分为局部变量和全局变量，全局变量中也称做成员变量，变量被定义在一个类中，且在所有的方法和函数之外时，局部变量存在于一个方法或者一个函数中。

（1）局部变量

局部变量，就是只在一个方法或者一个函数中起作用，超过这个范围，它将失去意义，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/PassTest.java”如下：

实例 1：演示局部变量

通过上面的讲解，用户可以看出，变量在程序中是随时可以改变的，随时都在传递着数据，下面通过一个实例进一步讲解局部变量的作用，其代码见“光盘：源代码/第 3 章/PassTest.java”，功能是分别计算三角形、正方形和长方形的面积，具体代码如下：

```
public class PassTest                                  //定义类 PassTest
{
  public static void main(String args[])
  {
        //三角形面积
        int a3=78,b3=56;                               //赋值 a3 和 b3
        int s3=a3*b3/2;                                //面积公式
        System.out.println("三角形的面积为"+s3);       //输出结果
        //正方形面积
        double a1=12.2;                                //赋值 a1
        double s1=a1*a1;                               //面积公式
        System.out.println("正方形的面积为"+s1);       //输出结果
      //长方形面积

        double a2=388.1,b2=332.3;                      //赋值 a2 和 b2
        double s2=a2*b2;                               //面积公式
        System.out.println("长方形的面积为"+s2);       //输出结果
```

```
        }
    }
```

对代码进行编译后，得到如图 3-2 所示的结果。

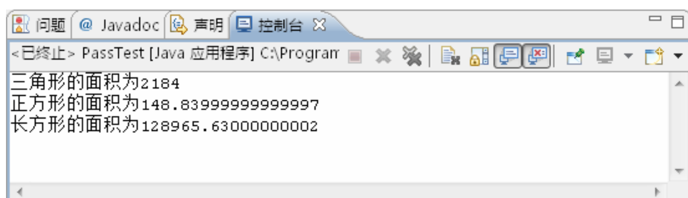
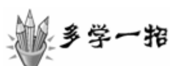


图 3-2 局部变量



通过上面的实例，读者认识了局部变量。下面给出一个程序，请读者朋友指出其中的局部变量，并指出它的作用范围，其代码见“光盘：源代码/第 3 章/bian.php”，具体代码如下：

```
public class Area
{
  int a1=7;
  int s1=a1*a1;
  public static void main(String args[])
  {
        //长方形面积
        double a2=7.1,b2=4.3;
        double s2=a2*b2;
        System.out.println("长方形的面积为"+s2);
       //三角形面积
        double a3=5.2,b3=6.7;
        double s3=a3*b3/2;
        System.out.println("三角形的面积为"+s3);
  }
}
```

在上面的代码中，S1、S2 和 S3 都是动态初始化代码，同时在初始化程序中还给出了 a1、a2、b2、a3 和 b3。对代码进行编译，运行后得到如图 3-3 所示的结果。



图 3-3 计算局部变量

（2）全局变量

理解了局部变量，再理解全局变量就容易得多，其实它相比局部变量就是作用范围变得更广了，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/Quan.java”，具体代码如下：

实例 2：演示局部变量的作用

通过上面的讲解，用户可以看出，变量在程序中是随时可以改变的，随时都在传递着数据，下面通过一个实例进一步讲解局部变量的作用，其代码见“光盘：源代码/第 3 章/Quan.java”，下面开始讲解具体代码，先定义变量 x，y，z，z1，a，b，c，d，e，具体代码如下：

```
public class Quan
{
    byte x;
      short y;                                    //定义变量 y
      int z;                                      //定义变量 z
      int z1;                                     //定义变量 z1
      long a;                                     //定义变量 a
      float b;                                    //定义变量 b
      double c;                                   //定义变量 c
      char d;                                     //定义变量 d
      boolean e;                                  //定义变量 e
```

然后设置 z1 的值，并分别输出 x，y，z，a，b，c，d，e 的值，具体代码如下：

```
    public static void main(String[] args)
    {
    int z1=111;                                //给 z1 赋值
       System.out.println("打印数据 z="+z1);     //下面开始分别输出数据
       Quan m=new Quan();
       System.out.println("打印数据 x="+m.x);
       System.out.println("打印数据 y="+m.y);
       System.out.println("打印数据 z="+m.z);
       System.out.println("打印数据 a="+m.a);
       System.out.println("打印数据 b="+m.b);
       System.out.println("打印数据 c="+m.c);
       System.out.println("打印数据 d="+m.d);
       System.out.println("打印数据 e="+m.e);
    }
}
```

对代码进行编译，运行后得到如图 3-4 所示的结果。

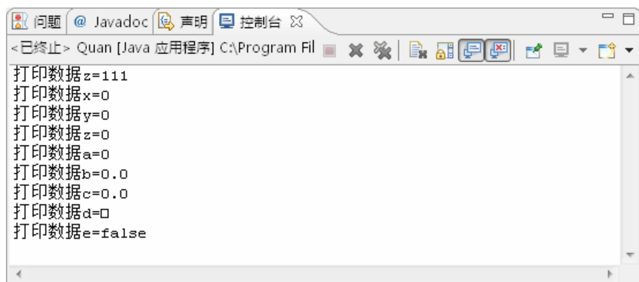
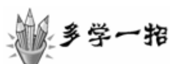


图 3-4　全局变量



通过上面的实例，使读者认识了全局变量，下面给出一个程序，请读者朋友指出其中的局部变量，并指出它的作用范围，其代码见“光盘：源代码/第 3 章/Quan1.php”，下面开始具体讲解，首先定义 String 并设置其值，代码如下：

```
public class Quan1                                    //定义类 Quan1
 {
   private static String staticValue = "";    //赋值 staticValue 为空
    @SuppressWarnings("unused")
    private String value;
            public Quan1() {
        super();
    }
       @SuppressWarnings("static-access")
```

然后在主程序中分别进行如下操作：

- 直接使用类名.静态方法名。
- 通过对象 t1 改变 static 变量。
- 通过对象 t2 改变 static 变量。
- 直接通过类名.静态变量存取。

对应代码如下所示：

```
public static void main(String []args)
 {
          //直接使用 类名.静态方法名()
          System.out.println("TestStatic.staticValue");
          Quan1.setStaticValue("static value");
            Quan1 t1=new Quan1();              //定义 Quan1 对象 t1
          Quan1 t2=new Quan1();              //定义 Quan1 对象 t2
                //t1, t2 存取到的值应该是一样的
        System.out.println("from t1:staticValue="+t1.getStatic  Value());
          System.out.println("from t2:staticValue="+t2.getStatic
Value());
          //通过对象 t1 改变 static 变量
          System.out.println("t1.staticValue");
          t1.setStaticValue("t1");
          //t1, t2 存取到的值应该是一样的
          System.out.println("from t1:staticValue="+t1.getStatic
Value());
          System.out.println("from t2:staticValue="+t2.getStatic
Value());
                //通过对象 t2 改变 static 变量
          System.out.println("t2.staticValue");
          t2.setStaticValue("t2");
```

```
                    //t1 受到影响
                    System.out.println("from t1:staticValue="+t1.getStatic
Value());
                }
                            public static String getStaticValue()
          {
                    return staticValue;
                }
                            public static void setStaticValue(String staticValue)
         {
                    //直接通过 类名.静态变量 存取
                  Quan1.staticValue=staticValue;
                }
                            public String accessStaticValue ()
         {
                    value=staticValue;   //普通的方法可以存取 static 方法或 static 变量
                                return staticValue;
                }
            }
```

对代码进行编译，运行后得到如图 3-5 所示的结果。

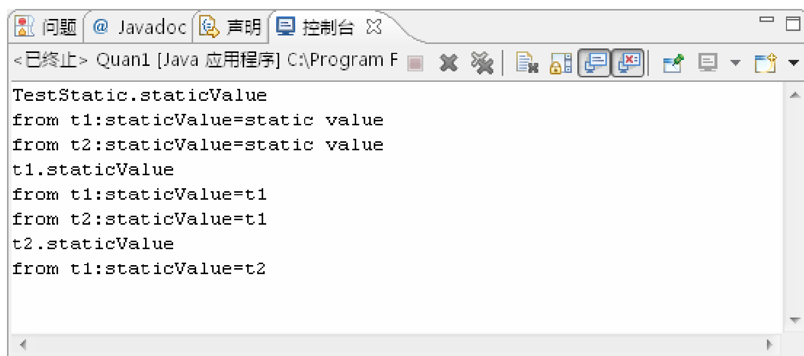


图 3-5　全局变量

3.2　数据类型

Java 中的数据类型可以分为简单数据和复杂数据两种类型。简单数据类型是 Java 的基础类型，它包括整数类型、浮点类型、字符类型和布尔类型，是本章的重点内容。复合数据类型是由简单数据类型组成的，是用户根据自己的需要定义并实现其运算的类型，如类、接口、数组，下面将对简单数据类型中最常用的几种数据类型进行讲解。

3.2.1　简单数据类型值范围

Java 中的简单数据类型是最简单的，主要由 byte、short、int、long、char、float、double

和 boolean 组成。简单数据类型所占的内存位数以及取值范围如表 3-1 所示。

表 3-1　数据类型

数 据 类 型	所 占 位 数	值 范 围
byte（字符类型）	8 位	–128~127
short（短整型）	16 位	–32768~32767
int（整型）	32 位	–2147483648~2147483647
long（长整型）	64 位	
float（单精度浮点型）	32 位	
double（双精度浮点型）	32 位	
char（字符型）	64 位	0~65535
boolean（布尔型）	1 位	True 或 false

3.2.2　字符型

代码 2：在 Java 中存储字符的数据类型是字符型，用 char 表示，下面引用一段代码，其代码见“光盘：源代码/第 3 章/Zifu.java”，具体代码如下：

```
public class Zifu
{
   public static void main(String args[])
   {
         char ch1='\u0001';                          //赋值 ch1
         char ch2='\u0394';                          //赋值 ch2
         char ch3='\uffff';                          //赋值 ch2
         System.out.println(ch1);                    //输出 ch1
         System.out.println(ch2);                    //输出 ch2
         System.out.println(ch3);                    //输出 ch2
   }
}
```

对代码进行编译，运行后得到如图 3-6 所示的结果。

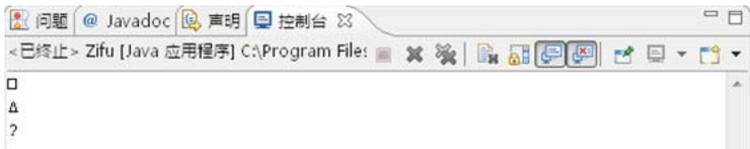


图 3-6　字符型

多学一招

上面的实例的运行结果是一些图形，为什么会出现这种情况呢？这是因为使用 Unicode 代码表示造成的，Unicode 所定义的国际化字符集支持表示迄今为止所有的字符集，如拉丁

文、希腊语等几十种语言，这些语言大部分是看不懂的，用户并不需要掌握。

读者需注意，在运行的结果处有一个问号，它可能是真的问号，也可能是不能显示的符号。那么究竟该怎样操作才能保证符号地正常输出呢？Java 提供了以“\”开头的转义字符（十六进制计数法用“\”和字母“U”开头，后面跟着十六进制数字）。常用的转义字符如表 3-2 所示。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/Zifu1.java”，具体代码如下：

```
public class Zifu1
{
   public static void main(String args[])
   {
         System.out.println("beijing\\shanghai");        //输出
         System.out.println("beijing\'shanghai");        //输出
         System.out.println("beijing\"shanghai");        //输出
         System.out.println("beijing\rshanghai");        //输出
   }
}
```

表 3-2　转义字符

转 义 字 符	描　　述	转 义 字 符	描　　述
\0x	八进制字符	\r	回车
\u	十六进制 Unicode 字符	\n	换行
\’	单引号字符	\f	走纸换页
\”	双引号字符	\t	横向跳格
\\	反斜杠	\b	退格

对代码进行编译，运行后得到如图 3-7 所示的结果。

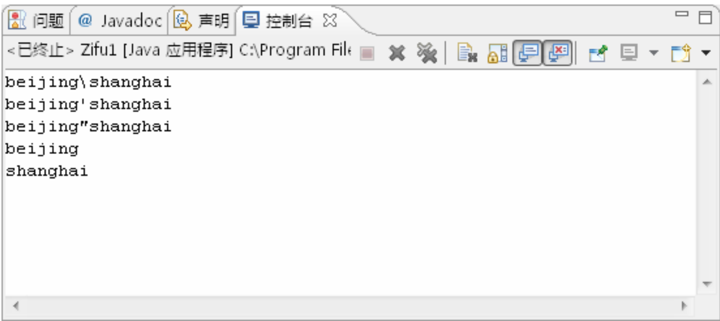


图 3-7　运行的结果

3.2.3　整型

整型是 Java 常用的数据类型，它是有符号的 32 位整数数据类型。整型 int 多用在数组、控制语句等地方。Java 系统会把 byte 和 short 自动提升为整型 int。

代码 3：下面通过一段代码讲解定义整型数据的方法，其代码见“光盘：源代码/第 3 章/Zheng.java”，具体代码如下：

```
public class Zheng                                         //定义类 Zheng
 {
        public static void main(String args[])
   {
        //正方形面积
        int b=7;                                           //赋值 b
        int L=b*4;                                         //赋值 L
        int s=b*b;                                         //赋值 S
        System.out.println("正方形的周长为"+L);            //输出周长
        System.out.println("正方形的面积为"+s);            //输出面积
        //三角形面积
        int a3=5,b3=7;                                     //赋值 a3 和 b3
        int s3=a3*b3/2;                                    //计算面积
        System.out.println("三角形的面积为"+s3);           //输出面积
   }
}
```

对代码进行编译，运行后得到如图 3-8 所示的结果。

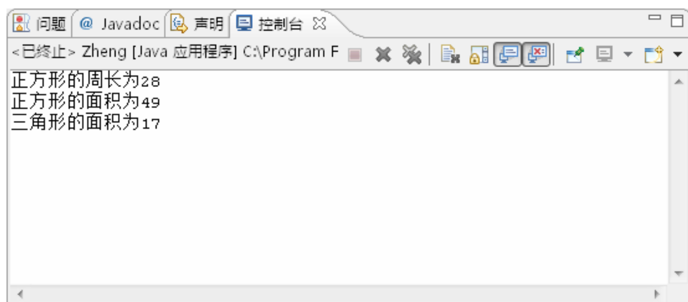


图 3-8　整型数据

3.2.4　浮点型

在 Java 编程过程中仅使用整型一个数据类型显然是不够的，还需要用到浮点型数据以支持大规模的程序开发。浮点型数据用来表示 Java 中的浮点数，其表示的主要是小数部分的数字。浮点型由两种类型组成，分别是单精度浮点型（float）和双精度浮点型（double），它们的值范围要比整型大许多，下面将对浮点型进行详细讲解。

（1）单精度浮点型

单精度浮点型是专指占用 32 位存储空间的单精度数据类型。不过在编程的过程中，只有在需要表示小数部分且对精度要求不高时，才会使用单精度浮点型，其他方面很少用到，故不做详细讲解。

（2）双精度浮点型

双精度浮点类型是专指占用类型 64 位存储空间的双精度数据类型，其在编程过程中使

用率极高，是一个能够为数值准确性提供最大保障的精确型数据类型。

代码 4：下面给出一段计算圆形面积的代码，其代码见“光盘：源代码/第 3 章/Syuan.java”，具体代码如下：

```
public class Syuan
 {
   public static void main(String args[])
   {
         double r=45.0324;                          //赋值 r
         final double PI=3.1415926;                 //PI
         double area=PI*r*r;                        //面积计算
         System.out.println("圆的面积是：S="+area);  //输出面积
   }
}
```

对代码进行编译，运行后得到如图 3-9 所示的结果。

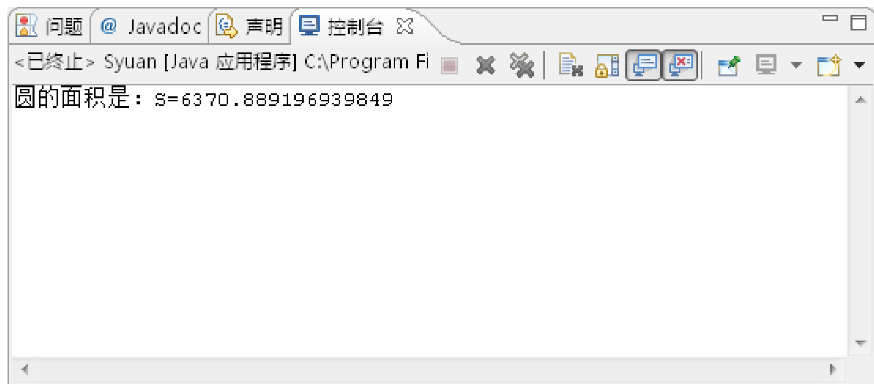


图 3-9　浮点型数据

提示：在上面的代码中多次出现“*”和“=”符号，它们是运算符。第一个符号相当于数学运算中的乘号，第二个是赋值符号，而非数学运算中的等号，在下一节中会讲到。

3.2.5　布尔型

布尔型是一种表示逻辑值的简单类型，它的值只能是 true 或 false 这两个值中的一个。它是所有诸如 a<b 之类的关系运算的返回类型。布尔型对治理像 if、for 这样的控制语句的条件表达式也是必需的。

代码 5：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/Bugu.java”，具体代码如下：

```
public class Bugu                                   //定义类
 {
   public static void main(String args[])
 {
         boolean b;                                 //定义 b
```

```
        b=false;                                        //赋值 b

        System.out.println("b is"+b);
        b=true;                                         //赋值 b
        System.out.println("b is"+b);
        if(b) System.out.println("This is executed.");
        b=false;                                        //赋值 b
        if(b) System.out.println("This is not executed.");
        System.out.println("10>9 is"+(10>9));
        }
}
```

对代码进行编译，运行后得到如图 3-10 所示的结果。

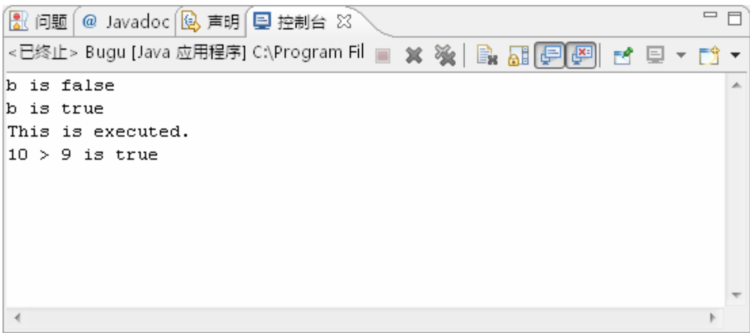


图 3-10　布尔类型

3.3　运算符

运算符是程序设计中重要的构成元素之一，运算符可以细分为算术运算符、位运算符、关系运算符、逻辑运算符和其他运算符。

3.3.1　算术运算符

算术运算符由基本算术运算符，取余运算符等主要用于算术表达式，其功能和用法与数学中加减乘除等的含义一样，如表 3-3 所示为算术运算符类型表。

表 3-3　算术运算符

类　型	运　算　符	说　明
基本运算符	+ – * /	加 减 乘 除
取余运算符	%	取余
递增或递减	++ – –	递增 递减

实例 3：演示算术运算符

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/yunsuan.java”如下：

```
public class yunsuan {
   public static void main(String args[]){
         int a=231;                          //赋值 a
         int b=4;                            //赋值 b
         System.out.println(a/b);            //下面开始分别输出对应的运算结果
         System.out.println(a+b);
         System.out.println(a-b);
         System.out.println(a*b);
         System.out.println(a%b);
         System.out.println(a++);
         System.out.println(a--);
         System.out.println(++a);
         System.out.println(--a);
   }
}
```

对代码进行编译，运行后得到如图 3-11 所示的结果。

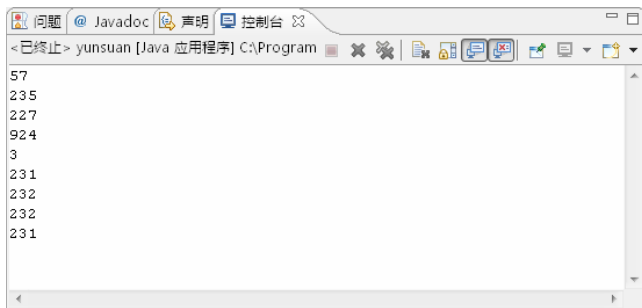


图 3-11　算术运算符

多学一招

算术运算符十分重要，初学者常容易混淆 a++和++a 的概念，单独情况下二者并没有区别，但作为表达式一部分时，前者先加 1 后执行程序，后者先执行程序后加 1，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/yunsuan1.java”如下：

```
public class yunsuan1
{
   public static void main(String args[])
   {
         int x=2000;                         //赋值 x
         int y=2000;                         //赋值 y
         int x1=2000;                        //赋值 x1
         int y1=2000;                        //赋值 y1
```

```
            System.out.println(x++);          //下面开始分别输出对应的运算结果
            System.out.println(++y);
            System.out.println(x1--);
            System.out.println(--y1);
        }
    }
```

对代码进行编译，运行后得到如图 3-12 所示的结果。

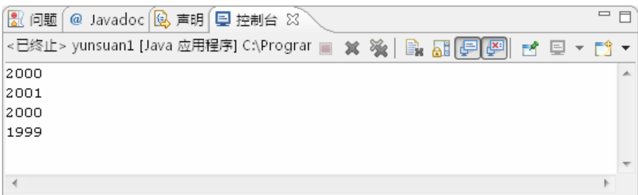


图 3-12　加减

3.3.2　关系运算符和逻辑运算符

在 Java 中，关系运算符和逻辑运算符十分重要，关系运算符是指值与值之间的相互关系，逻辑运算符是指可以用真值和假值链接在一起的方法，下面将分别讲解关系运算符和逻辑运算符。

（1）关系运算符

和数学运算一样，在编程中也有大于、小于、等于、不等于的关系，如表 3-4 所示列出了 Java 中的关系运算符类型，通过这些关系运算符执行程序，将会产生一个布尔值结果，即 True 和 False。在 Java 中任何类型的数据都可以用“==”符号比较是否相等，用“!=”符号比较是否不等。不过只有数字才能比较大小，但关系运算的结果可以直接赋予布尔变量。

表 3-4　关系运算符

类　　型	说　　明	类　　型	说　　明
==	等于	<	小于
! =	不等于	>=	大于等于
>	大于	<=	小于等于

实例 4：使用关系运算符

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/guanxi.java”如下：

```
public class guanxi {
    public static void main(String args[])
    {
            char a='k';                       //赋值 a
            char b='k';                       //赋值 b
            char c='A';                       //赋值 c
            int d=100;                        //赋值 d
```

```
            int e=101;                  //赋值 e
            System.out.println(a==b);   //下面开始分别输出对应的运算结果
            System.out.println(b==c);
            System.out.println(b!=c);
            System.out.println(d<e);
        }
    }
```

对代码进行编译，运行后得到如图 3-13 所示的结果。

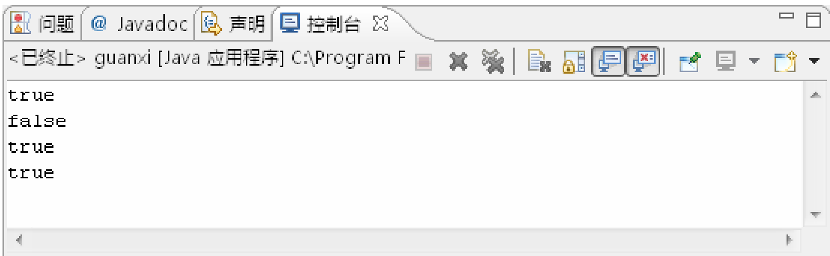


图 3-13　关系运算符

（2）逻辑运算符

布尔逻辑运算符是最常见的逻辑运算符，用于对布尔型数据进行布尔逻辑运算，Java 的布尔逻辑运算符如表 3-5 所示。

表 3-5　逻辑运算符

类　　型	说　　明	类　　型	说　　明
&&	与(AND)	|	简化或（Short-circuit OR）
||	或（OR）	&	简化并（Short-circuit AND）
∧	异或（XOR）	!	非（NOT）

逻辑运算符与关系运算符的结果一样，都是布尔值。

在 Java 程序设计中，“&&”和“||”布尔逻辑运算符并非总是对运算符右侧的表达式进行运算。如果使用“&”和“|”布尔逻辑运算符，则表达式的结果可以由运算符左侧的数据单独决定。关于表达式的内容将在下一节讲到。通过表 3-6，读者可以了解使用“&&”、“||”和“!”三种逻辑运算符号进行运算后的结果。

表 3-6　逻辑运算符运算结果

A	B	A&&B	A||B	!A
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

实例 5：使用逻辑运算符

在前面的讲解中，熟悉了逻辑运算符，也熟悉了逻辑运算符判断方法，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/luoji.java”如下：

```
public class luoji                                  //定义类
{
   public static void main(String[] args)
   {

         int a=10;                                  //赋值 a
         int b=12;                                  //赋值 b
         int c=111;                                 //赋值 c
         //下面开始分别输出对应的运算结果
         System.out.println(a>b||b<c);
         System.out.println(a>b&&b<c);
         System.out.println(!(a>b));
   }
}
```

对代码进行编译，运行后得到如图 3-14 所示的结果。

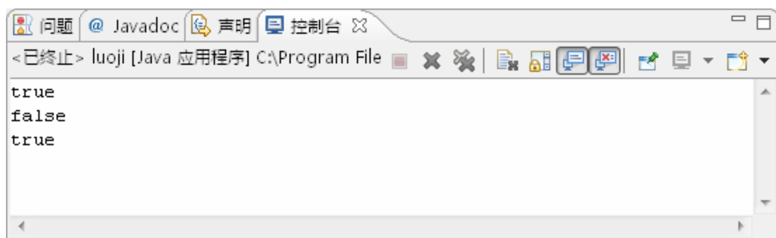


图 3-14　逻辑运算符

多学一招

通过上述实例读者可更加清楚地辨别不同逻辑运算符的运算方式。下面将给出一段代码，读者可以小试牛刀，根据表 3-6，自行判断出运算结果输入代码，编译并执行整个程序。其代码见“光盘：源代码/第 3 章/ luoji1.java”如下：

```
public class luoji1
 {
   public static void main(String[] args)
   {
         int d=101;                                       //赋值 d
         int k=121;                                       //赋值 k
         int j=123;                                       //赋值 j
         System.out.println(d<k||k<j);
         System.out.println(d<k||k>j);
         System.out.println(d>k||k>j);
         System.out.println(d>k||k<j);

         System.out.println(!(k>j));
   }
}
```

对代码进行编译，运行后得到如图 3-15 所示的结果。

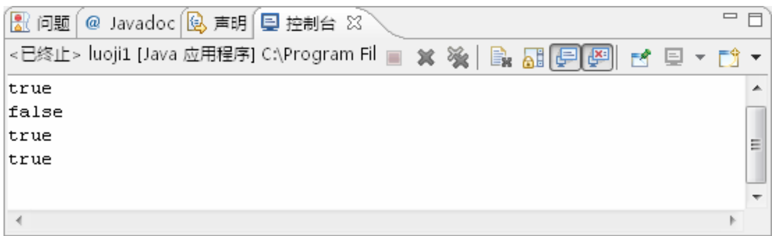


图 3-15　逻辑运行符

3.3.3　位运算符

在 Java 程序设计中，位运算符是用来对二进制数据进行操作的，位运算符细分为位逻辑运算符和移位运算符，如表 3-7 所示为位运算符。

表 3-7　位运算符

位逻辑运算符	说　　明	移位运算符	说　　明
~	按位取反运算	>>	右移
&	按位与运算	>>>	右移并用 0 填充
|	按位或运算	<<	左移
^	按位异或运算		

如表 3-8 所示是“操作数 A”和“操作数 B”按位逻辑运算后的结果。

表 3-8　位逻辑运算结果

操作数 A	操作数 B	A|B	A&B	A^B	~A
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

3.3.4　条件运算符

条件运算符是一种特殊的运算符，与前面所讲的几种运算符有很大不同。在 Java 程序设计中提供了一个三元运算符，它跟后面将要讲到的 if 语句有相似之处。条件运算符的目的是不同条件决定把不同的值赋予前面的变量，它的结构如下：

```
变量=（布尔表达式）？为 true 时所赋予的值：为 false 时所赋予的值；
```

代码 6：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 3 章/tiao.java”如下：

```
public class tiao
{
```

```
        public static void main(String args[])
        {
            double chengji=70;                           //赋值 chengji
            String Tiao=(chengji>=90)?"我已经很优秀了":"我现在不是很优秀了，我还
需要努力！";
            System.out.println(Tiao);                    //输出结果
        }
    }
```

对代码进行编译，运行后得到如图 3-16 所示的结果。

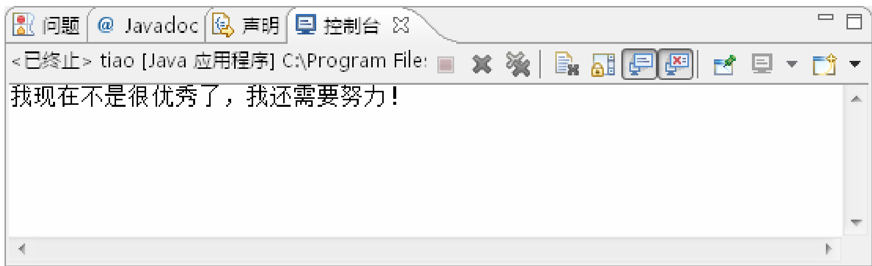


图 3-16　条件运算符

3.4　标识符和关键字

（1）标识符

标识符属于赋予类，在 Java 语言中，通常用标识符来识别类名、变量名、方法名、类型名、数组名和文件名。

标识符可由大小写字母、数字、下画线（_）、美元符号（$）组成，但不能以数字开头，且标识符没有最大长度限制。以下都是合法的标识符。

Chongqin_$　　D3Tf　　T_w_o　　$67.55

下面详细介绍判断标识符是否合法的方法：

- 标识符不能以数字开头，如 7788。
- 标识符中不能出现规定以外的字符，如 You’are、deng@qq.com。

标识符需要严格区分大小写，Java 中的 no 和 No 是完全不同的。另外还需注意的是，虽然使用“$”符号在语法上是被允许的，但编码规范中规定尽量避免使用它，容易混淆，造成不必要的麻烦。

（2）关键字

关键字是 Java 系统保留使用的标识符，是只有 Java 系统才能使用的标识符，用户和程序员不能使用。目前 Java 系统保留的关键字如表 3-9 所示。

表 3-9　Java 关键字

abstract	boolean	break	byte	case	catch	char	class	const	continue

default	do	double	else	extends	final	finally	float	for	goto
if	implements	import	instanceof	int	interface	long	nafive	new	package
private	protected	public	return	short	static	strictfp	super	switch	synchronized
this	throw	throws	transient	try	void	volatile	while	assert	

从表中可以看出 true、false 和 null，它们是 Java 定义的特殊值，虽然它们不是关键字，但也不能作为类名、方法名和变量名等使用。

3.5　疑难问题解析

本章详细介绍了量、数据类型、运算符等基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：在开发程序时定义了 int 类，为什么程序提示“int”有错误？

解答：在 Java 程序中，int 是保留字符，用来表示整型数据类型。凡是保留的字符，都不可定义为一个名称，如变量名、常量名和类名。

读者疑问：Java 程序在运算时，能否像其他语言一样，用不同的数据类型进行计算？比如 char a 与 int c 是否可以相加？

解答：可以。在 Java 程序中经常遇到这种情况，两种不同的数据类型进行计算，成功与否取决于相加的数据类型是否兼容。



职场点拨——不同的客户，不同的处理方式

程序员并非只负责技术方面的工作，当项目做完之后，客户可能有不满意的地方，有很多问题是产品部的同事们不能解决的，这个时候就需要程序员提供支持。在面对我们的“上帝”——客户时，要用最合理的沟通手段来促成合作。为了帮助大家更好地应对客户，在这里将列出 4 条和客户沟通的技巧。

1）对比法：通过将自己和其他同行进行有效对比，在效果、价格等方面体现出较为明显的优势，让客户真正产生兴趣，从而达成合作。

2）举例法：在开发过程中，经常遇到客户怀疑合作后是否能够开发出满意系统的问题。此时可以用举例法，用他身边或他自己有切身感受的例子就可以达到很好的效果，使客户打消疑虑，快速做出积极的决定。

3）避实就虚法：如果客户对项目表示出了浓厚的兴趣，我们就要运用专业的知识和灵活的沟通技巧促使客户达成合作，专业的知识更具有说服力。如果客户因为自身原因或者对你推荐的方案不感兴趣，则应该立刻停止关于该方案话题的沟通，转向客户比较感兴趣的话题。甚至要立即停止沟通，找其他合适的机会再说，避免和客户关系的进一步恶化。

4）围魏救赵法：从客户身边的人入手，通过这些客户的关系人来间接影响客户本身，从而达到与客户合作的目的。

第 4 章　字符串、运算符和表达式

在 Java 程序中，运算符、表达式和字符串是三种不可或缺的重要元素。运算符的种类有很多种，如算术运算符、位运算符、关系运算符、逻辑运算符等，将操作数和运算符连接起来，就组成了一个符合 Java 语法规则的表达式。运算符和表达式都是程序的基础，在本章中将为读者一一讲解。本章主要内容如下：

- ❑ 运算符。
- ❑ 表达式。
- ❑ 字符串。
- ❑ 职场点拨——提高你的职场生存方式。

2010 年 X 月 X 日，天气阴

上班 3 个月了，我想买笔记本电脑，也想买高档手机，可是只攒了几千块钱。



一问一答

小菜：“我好想买一台高配置的笔记本，再买一部高档手机，可惜攒的钱远远不够！”

Wisdom：“呵呵，你挺会享受的，但追求高品位的生活也未必一定要买笔记本和高档手机啊！”

小菜：“……”

Wisdom：“建议你买一台组装机，物美价廉，用来学习编程绰绰有余。至于手机，可以先买部普通的，等下半年有钱了再买高档手机。”

小菜：“嗯，言归正传，本章所学的运算符和表达式有什么用呢？”

Wisdom：“本章我们要讲的运算符和表达式是一种方式，是一个对程序进行处理的处理方式。通俗一点讲，运算符和表达式就是加减乘除之类的运算符号，需要使用加法时就‘+’，需要使用减法时就 ‘–□。在本章最后，还将介绍优化职场生存方式的方法！”

4.1　再看运算符

在 Java 中，运算符的重要性不言而喻。如果没有运算符，Java 程序基本上没有作用。运算符大致可以细分为算术运算符、位运算符、关系运算符、逻辑运算符和其他运算符，下面对其进行讲解。

4.1.1　算术运算符

算术运算（Arithmetic Operators）符，就是用来处理数学运算的符号，是最简单、最常用的符号。但凡是针对数字的处理几乎都会用到算术运算符号。算术运算符可以分为几大类，基本运算符、取模运算符和递增或递减运算符，如表 4-1 所示。

表 4-1　算术运算符

类　　型	运　算　符	说　　明
基本运算符	+ – * /	加 减 乘 除
取模运算符	%	取模
递增或递减	++ – –	递增 递减

（1）基本运算符

在 Java 程序中，基本运算符的应用十分广泛，下面通过一个实例进行讲解。

实例 6：使用基本运算符实现加减乘除 4 种运算

通过下面的这个实例，使读者初步了解 Java 基本运算符的加减乘除 4 种运算，其代码见“光盘：源代码/第 4 章/JiBen1.java”：

```
public class JiBen1
{
   public static void main(String args[])
   {
         int a=12;
         int b=4;
//运算符
         System.out.println(a-b);
         System.out.println(a+b);
         System.out.println(a*b);
         System.out.println(a/b);

   }
}
```

编译并运行上面这段代码，得到如图 4-1 所示的结果。

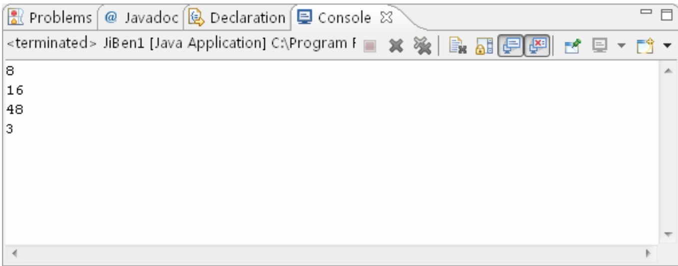
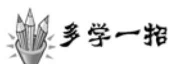


图 4-1　执行后的结果



在基数运算的过程中，用户一定要注意整数相除只保留整数部分，其小数不会保留。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/Jiben2.java”：

```
public class Jiben2 {
   public static void main(String args[])
   {
         int a=126;
         int b=50;
         int c=700;
         System.out.println(a/b);
         System.out.println(c/b);
         System.out.println(a/b+c);
         System.out.println(a+b/c);
               }
}
```

运行代码，得到如图 4-2 所示的结果。

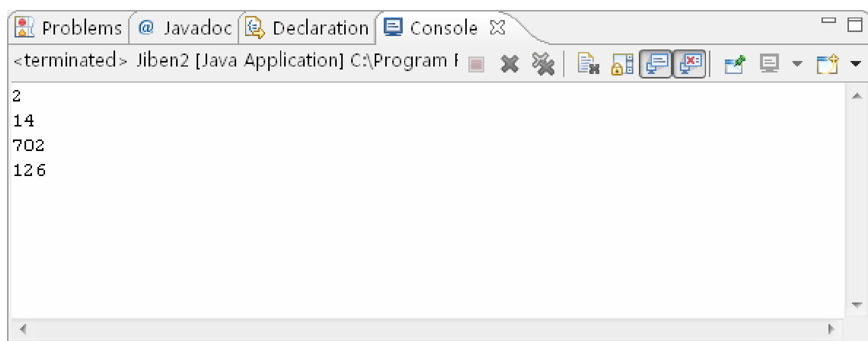


图 4-2　基本运算的结果

提示：电脑运算和数学运算有些不同，分母在有些时候不能为零，为零即导致程序错误。不过有时程序的分母为零并不是错误，其代码见“光盘：源代码/第 3 章/Jiben.java”：

```
public class Jiben {
   public static void main(String args[])
   {
         int AAA=126;
//整形数据分母不能为零
         System.out.println(a/0);
               }
}
```

运行代码，得到如图 4-3 所示的结果。

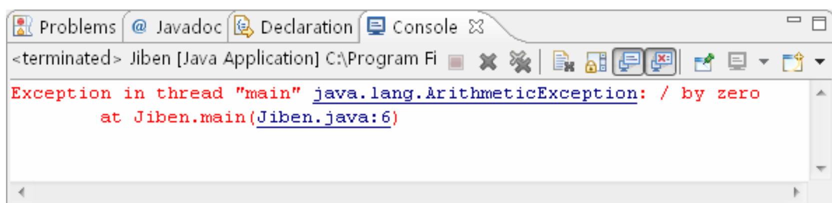


图 4-3　运行结果

上面的结果提示用户分数的分母不能为零，现在对将这段程序稍做修改，将“int AAA = 126”改为“double AAA=126”，运行后会得到图 4-4 所示的结果。

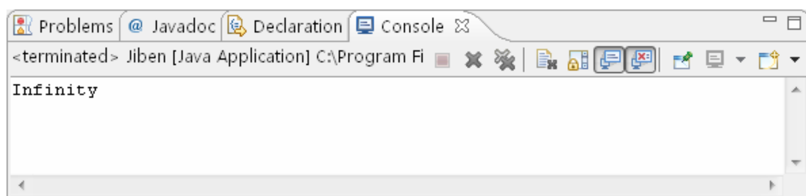


图 4-4　更改后的结果

该结果说明，在基本运算符中，只有将分子定义为 double 型时，分母为零是正确的，运算得到的值是无穷大，这一点希望初学者加以理解。

（2）求余运算符

求余运算符是一种特殊的运算符，在数学中也很少用到。其实求余运算符很容易理解，它一般用在除法中，它取的值不是商，而是取余数，如 5/2，它取的是余数 1，而不是商值 2。下面通过一个实例进行讲解。

实例 7：使用“%”运算符

通过下面的这个实例，使读者明白“%”运算符的作用，其代码见“光盘：源代码/第 4 章/Yushu.java”：

```
public class Yushu
 {
   public static void main(String[] args) {
//求余数
         int A=19%3;
         int K=-19%-3;
         int Q=19%-3;
         int J=-19%3;
         System.out.println("A=19%3 的余数"+A);
         System.out.println("K=-19%-3 的余数"+K);
         System.out.println("Q=19%-3 的余数"+Q);
         System.out.println("J=-19%3 的余数"+J);
   }
   }
```

运行代码，得到如图 4-5 所示的结果。

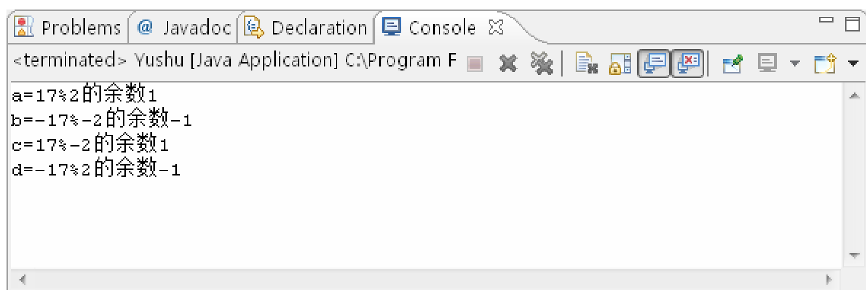


图 4-5　取模

多学一招

求余运算是计算机中独有的运算方式，求余运算遵循一个规律，x%y = x%|y|，上面的代码得到的结果就是遵循这个规律所得到的，下面再通过一段代码进行测试，其代码见“光盘：源代码/第 3 章/Yushu1.java”：

```
public class Yushu1
 {
         public static void main(String[] args)
{
               int x=37%5;
               int y=17%-5;
               int z=-57%-5;
               int m=-37%5;
               System.out.println("x=1237%5 的余数"+x);
               System.out.println("y=117%-5 的余数"+y);
               System.out.println("z=-517%-5 的余数"+z);
               System.out.println("m=-317%5 的余数"+m);
         }
   }
```

运行代码，得到如图 4-6 所示的结果。

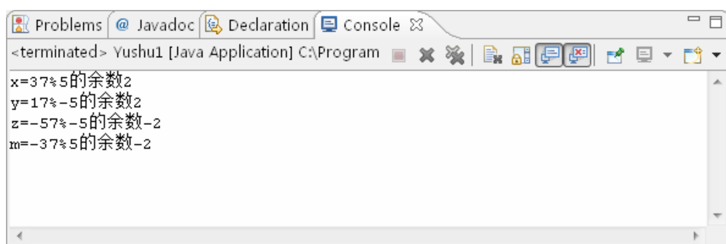


图 4-6　取模规则

（3）递增递减

“++”、“– –”每执行一次，变量将会增加 1 或者减少 1，它可以放在变量的前面，也可以放在变量的后面，无论哪一种都能改变变量的结果，但它们有一些不同，这种变化让初学

程序的人也甚感疑惑。递增、递减对于刚学程序的人来说是一个难点，读者朋友一定强加理解，理解的不是++与– –的问题，而是在变量前用还是变量后用的问题。下面通过一个实例进行讲解。

实例 8：使用递增递减运算符

本实例代码见“光盘：源代码/第 3 章/Dione.java”：

```
public class Dione{
   public static void main(String args[])     {
         int a=199;
         int b=1009;
//数据的递增与递减
         System.out.println(a++);
         System.out.println(a);
         System.out.println(++a);
         System.out.println(b--);
         System.out.println(b);
         System.out.println(--b);
   }
}
```

运行代码，得到如图 4-7 所示的结果。

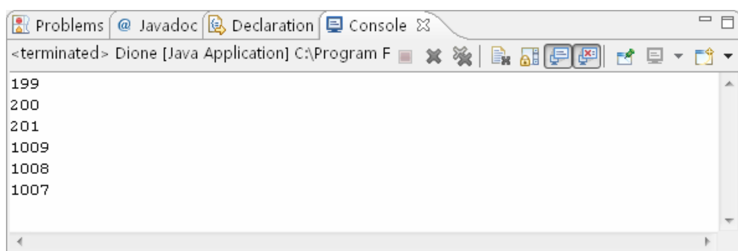


图 4-7　执行的结果

在上面程序中，a++是先执行程序再加 1，++a 是先加 1 再执行程序，b--是先执行 b 的值再减 1，--b 是 b 先减 1，再执行程序。比如，“system.out.println(a++);”它是先执行再加 1，所以它输出的值应该是初始值 199，“system.out.println(a);”因为前一句代码将其加 1，所以结果有所变化，为 200，“system.out.println(++a)”是先加 1，其结果就是 2001，而后面的代码相信读者也应该懂了。

多学一招

下面通过一段代码巩固递增递减知识，其代码见“光盘：源代码/第 5 章/DiTwo.java”：

```
public class DiTwo
 {
   public static void main(String args[])
   {
         int x=2;
```

```
            int y=3;
            int x1=7;
            int y1=8;
            System.out.println(x1+x++);
            System.out.println(++y-x);
            System.out.println(x1---y);
            System.out.println(--y1*y);
        }
    }
```

运行代码，得到如图 4-8 所示的结果。

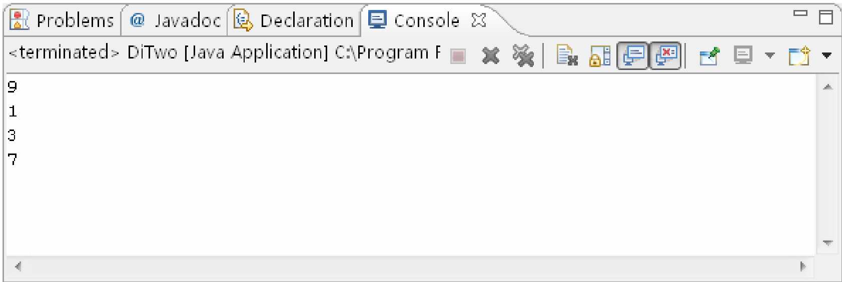


图 4-8　递增递减

4.1.2　关系运算符和逻辑运算符

在程序设计中，关系运算符（Relation Openator）和逻辑运算符（Logical Operator）十分重要，关系运算符是指值与值之间的相互关系，而逻辑（Logical）关系是指可以用真值和假值连接在一起的方法。因此关系运算符产生的结果可以是真或假，它们经常与逻辑运算符一起使用。

（1）关系运算符

关系运算符是指对两个表达式进行比较，返回一个真或假值。如表 4-2 列出了 Java 中的关系运算符，通过这些关系运算符运算产生的结果是一个布尔值，即 True 和 False。在 Java 中任何类型的数据都可以用“= =”进行比较是否相等，用“!=”比较不等，但只有数字才能比较大小。关系运算的结果可以直接赋予布尔变量。

表 4-2　关系运算符

类　　型	说　　明	类　　型	说　　明	类　　型	说　　明
= =	等于	>	大于	>=	大于等于
! =	不等于	<	小于	<=	小于等于

代码 7：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/guanxi.java”：

```
    public class guanxi
    {
        public static void main(String args[])
```

```
    {
        char a='D';
        char b='K';
        int c=63;
        System.out.println(a>b);
        System.out.println(a>c);
    }
}
```

运行代码，得到如图 4-9 所示的结果。

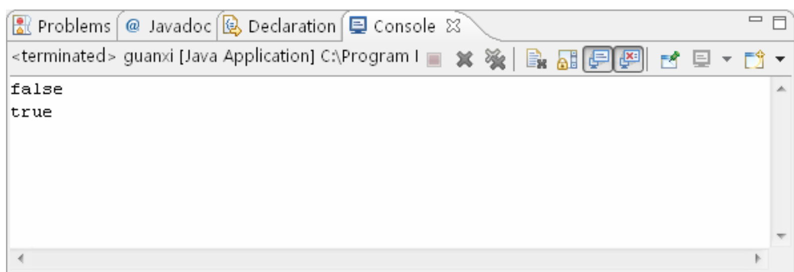


图 4-9　关系运算符

（2）逻辑运算符

逻辑运算符的结果是真或者假，布尔逻辑运算符是最常见的运算符。表 4-3 所示是操作数 A 和操作数 B 按位逻辑运算的结果。

表 4-3　逻辑运算符

A	B	A&&B	A||B	!A
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

代码 8：下面通过一段代码进行讲解，以加深读者对关系运算符和逻辑运算符的理解，其代码见“光盘：源代码/第 4 章/Luoji.java”：

```
public class Luoji
{
    public static void main(String[] args)
    {
        int d=10;
        int k=12;
        int j=111;
        System.out.println(d<k||k<j);
        System.out.println(d>k&&k<j);
        System.out.println(!(d>k));
    }
}
```

运行代码，得到如图 4-10 所示的结果。

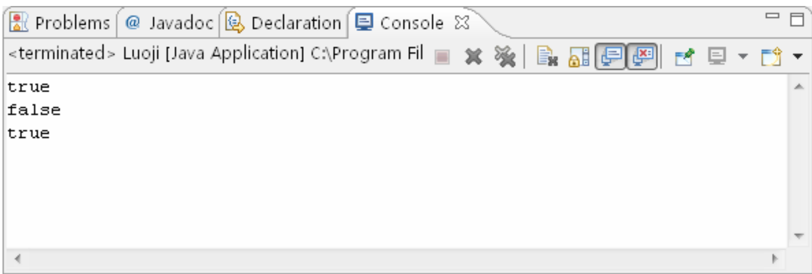


图 4-10　逻辑运算符

4.1.3　位运算符

位运算（bitwise operators）可以直接对整数类型的位进行操作，这些整数类型包括 long、int、short、char 和 byte，如表 4-4 所示。

表 4-4　逻辑运算符

位逻辑运算符	说　　明	位逻辑运算符	说　　明
~	按位取反运算	>>	右移
&	按位与运算	>>>	右移并用 0 填充
|	按位或运算	<<	左移
^	按位异或运算		

既然位运算符能够在整数范围内对位操作，那么这样的操作对一个值产生什么效果是很重要的。具体来说，知道 Java 是如何存储整数值并且如何表示负数是非常有用的，通过上面的操作，对应的结果如表 4-5 所示。

表 4-5　位逻辑运算结果

操作数 A	操作数 B	A|B	A&B	A^B	~A
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

下面讲解移位运算符，移位运算符通过把数字的位向右或向左移动，从而产生一个新数字，需要注意的是移位运算符只能用在整数型上，不能用在浮点型上。要确定移位的结果，必须先将数字转换为二进制，然后再进行移位。

代码 9： 下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/Wei.java”：

```
public class WeiOne
{
```

```
        public static void main(String args[])
        {
            String binary[]={"0000", "0001", "0010", "0011", "0100",
"0101", "0110", "0111", "1000", "1001", "1010", "1011", "1100", "1101", "1110",
"1111"
            };
            int a=3; //0+2+1 or 0011 in binary
            int b=6; //4+2+0 or 0110 in binary
            int c=a|b;
            int d=a&b;
            int e=a^b;
            int f=(~a&b)|(a&~b);
            int g=~a&0x0f;
            System.out.println("a="+binary[a]);
            System.out.println("b="+binary[b]);
            System.out.println("a|b="+binary[c]);
            System.out.println("a&b="+binary[d]);
            System.out.println("a^b="+binary[e]);
            System.out.println("~a&b|a&~b="+binary[f]);
            System.out.println("~a="+binary[g]);
            }
        }
```

使用位逻辑运算符“^”运算时，只有在比较两个位不相等的操作数时结果才返回 1，否则结果是零。运行代码，得到如图 4-11 所示的结果。

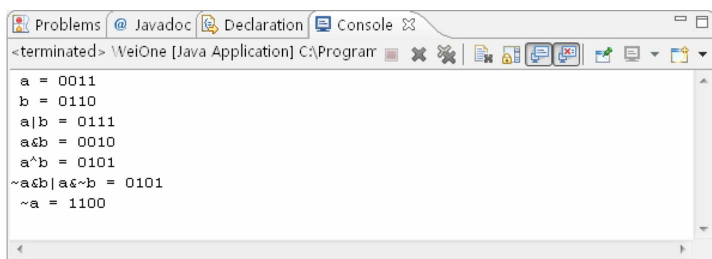


图 4-11　位运算符

4.1.4　条件运算符

条件运算符与其他的运算符有一定的不同，在 Java 程序里，条件运算符与后面将要讲到的 if 语句和 Switch 语句十分相似，条件运算符的目的是决定把哪儿的值赋给前面的变量，其基本结构如下：

```
变量=（布尔表达式）？为 true 时所赋予的值：为 false 时所赋予的值；
```

实例 9：使用条件运算符

条件运算符的格式大家已经清楚，它会根据条件的不同而赋值。下面看一个使用条件运

算符的实例，其代码见“光盘：源代码/第 4 章/ Dione.java”：

```
public class DosEquis
{
public static void main(String[] args)
{
        char x='X';
        int i=0;
        System.out.println(true ? x : 0);
        System.out.println(false ? i : x);
    }
}
```

运行代码，得到如图 4-12 所示的结果。

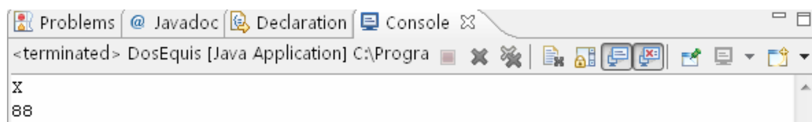
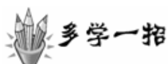


图 4-12　条件运算符



上面这个程序由两个变量声明和两个 print 语句构成。第一个 print 语句计算条件表达式(true ? x : 0)并得出结果，这个结果是 char 类型变量 x 的值‘X’。而第二个 print 语句计算表达式(false ? i : x)并得出结果，这个结果依旧是 x 的值‘X’，因此这个程序的最终结果应该是 XX。然而运行程序却发现其结果是 X88。也就是说第一个 print 语句的结果是 X，而第二个却是 88，它们的不同行为说明了什么呢？这是因为在这两个表达式中，每一个表达式的第二个操作数的类型都不相同，x 是 char 类型的，而 i 都是 int 类型的，计算混合类型会引起混乱。

4.1.5　赋值运算符

赋值运算符是一个等号“=”，它在 Java 中的运算与在其他计算机语言中的运算一样，起到赋值的作用。其通用格式为：

```
var=eXPression;
```

其中，变量 var 的类型必须与表达式 eXPression 的类型一致。

赋值运算符有一个有趣的属性，它允许我们对一连串变量赋值。请看下面的例子：

```
int x, y, z; x=y=z=100;        // set x, y, and z to 100
```

该例子使用一个赋值语句对变量 x、y、z 都赋值 100。原理是因为“=”运算符产生右边表达式的值，因此 z = 100 的值是 100，然后该值被赋给 y，并依次被赋给 x。使用“字符串赋值”是给一组变量赋予同一个值的简单办法。在赋值时，类型必须匹配。

代码 10：下面通过一段代码讲解赋值运算符的用法，其代码见“光盘：源代码/第 4 章/ fuzhi.java”：

```
public class fuzhi
 {
   public static void main(String args[])
   {
//定义的字节数据
         byte a=9;
         byte b=7;
         byte k=a+b;
         System.out.println(c);
   }
}
```

运行后提示类型不匹配，赋值运算符主要用于数据类型计算，在赋值时，类型要匹配，上面程序运行得到如图 4-13 所示的结果，提示用户赋值时数据类型不匹配造成错误。

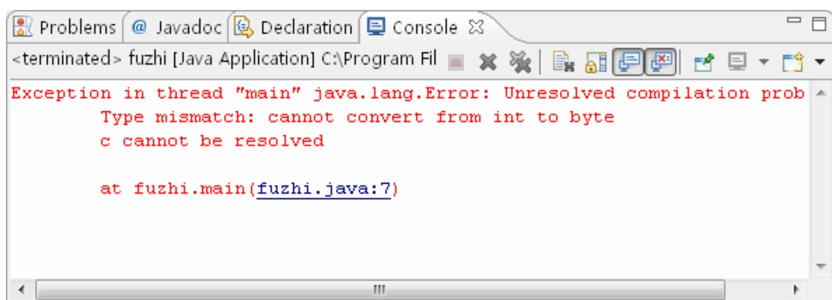


图 4-13　数据类型不匹配

4.2　表达式

将相同类型的数据（如常量、变量、函数等）用运算符号按一定的规则连接起来所组成的有意义的式子称为表达式。在 Java 程序中，表达式往往是以与运算符结合的形式出现的，本节将重点讲解运算符与表达式的结合。

4.2.1　什么是表达式

在前面的章节中，读者朋友多多少少都接触了一些表达式，但是对表达式的概念还比较模糊，下面将深入讲解表达式。

在程序代码内部，每个表达式都有其对应的数据类型，例如 int n = 10; 和 int m = 10; 都是带着数据类型的表达式。在实际的程序代码中，大部分的表达式不能单独作为代码中的一行，否则程序会提示语法错误，例如：

```
int a=10;
int b=20;
a+b;                          //不能单独成行
```

在表达式中，能够单独成行的运算符包括赋值运算符和递增、递减运算符。

4.2.2　表达式的优先级

“优先级”简单来讲，就是“规定哪个语句块先执行，哪个语句块后执行”。Java 程序有着严格的优先级规则，它和代数中先乘除后加减的原则一样，按优先级从高到低进行运算，如表 4-6 所示。

表 4-6　运算符的优先级

优　先　级	运　算　符	解　　释
1	[ ]　()	分隔符
2	++　——　~ !	递增减运算 按位取反 逻辑非
3	*　/　%	算术乘除运算
4	+　–	算术加减运算
5	>>　<<　>>>	移位运算
6	<=　<　>　>=	大小关系运算
7	==　! =	相等关系运算
8	&	按位与运算
9	^	按位异或运算
10	|	按位或运算
11	&&	布尔逻辑与运算
12	||	布尔逻辑或运算
13	? :	条件运算
14	=	赋值运算

4.2.3　表达式的应用

Java 虚拟机对表达式的求值是按照从左到右的顺序进行的，并遵循标准的运算优先级法则。当然，与其他语言一样，求值顺序还可以用它的优先级来控制，下面通过一个实例进行讲解。

实例 10：使用表达式常与运算符

表达式常与运算符结合起来进行应用，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/Biaoone.java”：

```
public class Biaoone
{
   public static void main(String args[])
   {
   int a=231;
   int b=4;
   int h=56;
   int k=45;
   int x=a+h/b;
   int y=h+k;
   System.out.println(x);
```

```
System.out.println(y);
System.out.println(x==y);
  }
}
```

运行程序，得到如图 4-14 所示的结果。

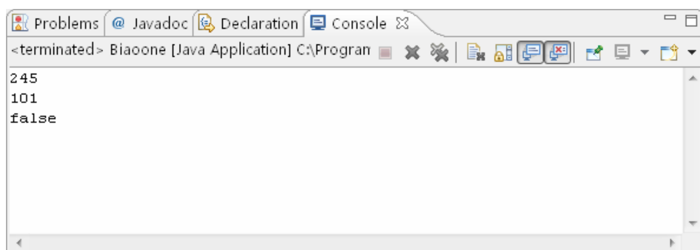
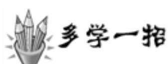


图 4-14　表达式



Java 编程语言中由算术运算符组成的表达式是按照先乘除，再加减的规则运算的。混合运算同样遵循优先级法则，比如算出 X，算出 Y，然后比较 X==Y 是否相等，最后得到 false 结果。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/biaotwo.java”：

```
public class biaotwo
{
public static void main(String[] args)
{
        int aa=231;
        int bb=4;
      int hh=56;
      int kk=45;
      int xx=aa-bb+kk;
      int yy=bb*kk/bb+hh;
    System.out.println(xx);
    System.out.println(yy);
    System.out.println(xx!=yy);
  }
}
```

运行代码，得到如图 4-15 所示的结果。

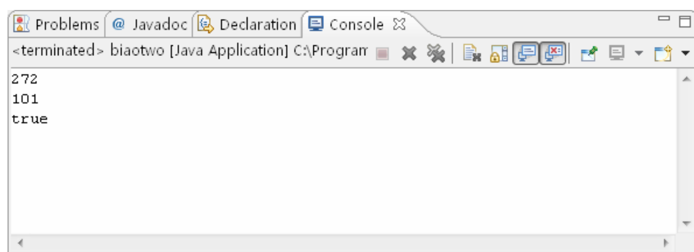


图 4-15　表达式

4.3 字符串

字符串（String）是由零个或多个字符组成的有限序列，一般记为 s='a1a2…an'(n>=0)。它是编程语言中用于表示文本的数据类型，通常以串的整体作为操作对象，如在串中查找某个子串、求取一个子串、在串的某个位置上插入一个子串以及删除一个子串等。两个字符串相等的充要条件是：长度相等，并且各个对应位置上的字符都相等。假设 p、q 是两个串，求 q 在 p 中首次出现的位置的运算叫做模式匹配。串的两种最基本的存储方式是顺序存储方式和链接存储方式。

4.3.1 字符串的初始化

在 Java 程序中，使用 new 关键字创建 String 的结构如下：

```
String a=new String();
```

在上面的这段代码中，用户就创建了一个 String 类，并把它赋给变量，但它还是一个空的字符串，接下来就要为这个字符串赋值：

```
A=”I am a person Chongqing”
```

在程序中，用户可以将两句代码合并，就可以产生一种简单的字符串表示：

```
String s=new String（”I am a person Chongqing”）;
```

除了上面的表示方法，还有另外一种字符串表示方式：

```
String s=（”I am a person Chongqing”）;
```

代码 11：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/Stringone. java”：

```
public class Stringone
{
   public static void main(String[] args)
{
         String str="清风明月路";
         System.out.println("白霜");
        String cde="甘泉石上流";
        System.out.println(str + cde);
   }
}
```

运行代码，得到如图 4-16 所示的结果。

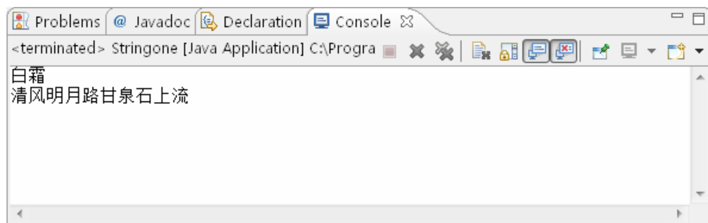
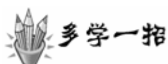


图 4-16 字符串



在 Java 程序设计里，字符串可以使用不同的方法进行初始化，然后将两个字符相加。下面通过一段代码讲解字符串的初始化，读者可根据程序判断结果，然后将代码输入电脑，进行保存，让读者对字符串的初始化更上一层楼。其代码见“光盘：源代码/第 4 章/Stringtwo.java”：

```
public class Stringtwo
{
   public static void main(String[] args)
{
         String A=new String("明月几时有，");
         String  K="把酒问青天，";
         String  Q="不知天上宫阙，";
         String  J="今夕是何年。";
       String E=A+K+Q+J;
       System.out.println(A);
       System.out.println(K);
       System.out.println(Q);
       System.out.println(J);
       System.out.println(E);
   }
}
```

提示：字符串并不是原始的数据类型，而是复杂的数据类型。对字符串进行初始化的方法不只一种，但也没有规定谁更优秀，用户可以根据自己的习惯选择使用。

运行代码，得到如图 4-17 所示。

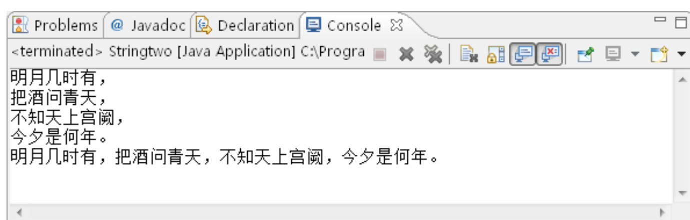


图 4-17　字符串的初始化

4.3.2　String 类

在 Java 程序里，String 类是用来操作字符串的，其中有许多操作方法供编程者使用，下面将讲解一些常用的方法。

（1）索引

在 Java 程序设计里，有一种方法可以返回 String 指定索引的位置，用户需要注意的是，它的数字是从零开始，其格式如下：

```
public char charAt (int index)
```

代码 12：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/suoyin.Java”：

```
public class suoyin
{
   public static void main(String args[])
   {
         String x="dongjiemeili";
         System.out.println(x.charAt(5));
   }
}
```

运行代码，得到如图 4-18 所示。

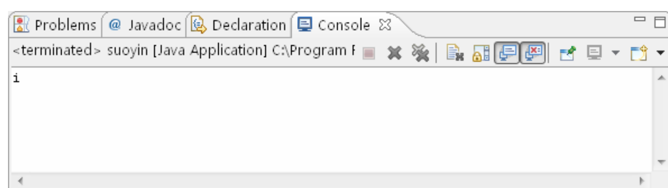


图 4-18　索引

（2）追加字符串

追加字符串，顾名思义是在字符串的末尾再添加字符串。追加字符串是一种常用的方法，使用起来十分简单，其格式如下：

```
Public String concat (String S)
```

代码 13：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/zhuijia.java”：

```
public class zhuijia
 {
   public static void main(String args[])
   {
         String x="2009，世界经";
         System.out.println(x.concat("济，在慢慢地复苏……"));
   }
}
```

运行代码，得到如图 4-19 所示的结果。

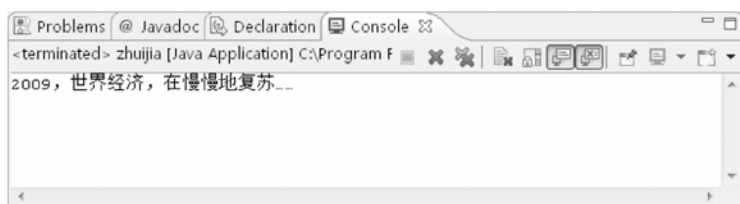


图 4-19　追加字符串

（3）比较字符串

比较字符串是比较两个字符串是否相同，如果相同则返回一个值 true，如果不相同则返回一个 false 值，其格式如下：

```
public Boolean equalsIgnoreCase(String s)
```

代码 14：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/bijiao.java”：

```
public class bijiao
{
   public static void main(String args[])
    {

      String x="student";
      String xx="STUDENT";
      String y="student";
      String z="T";
      System.out.println(x.equalsIgnoreCase(xx));
      System.out.println(x.equalsIgnoreCase(y));
      System.out.println(x.equalsIgnoreCase(z));
    }
}
```

运行代码，得到如图 4-20 所示的结果。

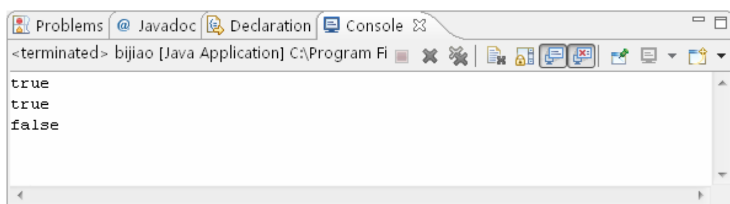


图 4-20　比较字符串

（4）获得求字符串长度

在 String 方法中，有一种可以求出字符串长度的方法，其格式如下：

```
public int length ()
```

代码 15：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/Qiuchang. Java”：

```
public class Qiuchang
{
   public static void main(String args[])
   {
         String x="hello";
         String y="浙江横店镇";
         String yy="长江三峡好地方";
```

```
        String z="my heart will go on";
    //计算数据的长度
        System.out.println(x.length());
        System.out.println(y.length());
        System.out.println(yy.length());
        System.out.println(z.length());
      }
    }
```

运行代码，得到如图 4-21 所示的结果。

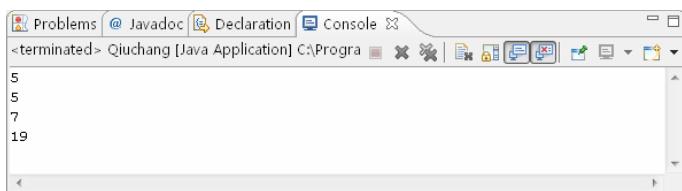


图 4-21　获得求字符串长度

（5）替换字符串

实际上替换是两个动作，第一个是查找，第二个是替换，使用它替换字符串十分简单，用户只需要记住这样一个格式就可以了，其格式如下：

```
public String replace（char old，char  new）
```

代码 16： 下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/Tihuan.java”：

```
public class Tihuan
{
    public static void main(String args[])
      {
            String x="我想我得去了";
           //替换字符
                    String y=x.replace('走','去');
                    System.out.println(y);
      }
}
```

运行代码，得到如图 4-22 所示的结果。

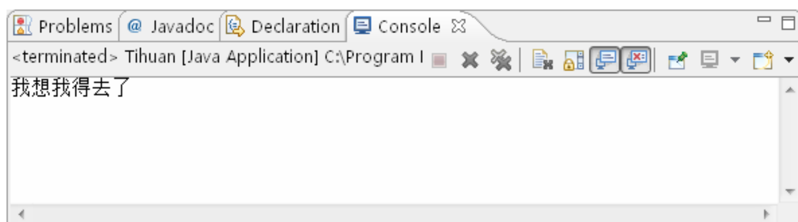


图 4-22　替换字符串

（6）字符串的截取

在编程过程中常需要从冗长的字符串中截取其中的一段，其格式如下：

```
public String substring（int begin）
```

另一种方法格式如下：

```
public String substring（int begin，int end）
```

代码 17：上面两个方法都可以截取字符串，下面通过一段代码讲解这两种方法，其代码见“光盘：源代码/第 4 章/Jiequ.java”：

```
public class Jiequ
{
   public static void main(String args[])
   {
         String x="杭州，有着悠久的历史，尤其是西湖，闻名中外。";
         //截取字符串
         String y=x.substring(6);
         String z=x.substring(6,16);
         System.out.println(x);
         System.out.println(y);
         System.out.println(z);
   }
}
```

运行代码，得到如图 4-23 所示的结果。

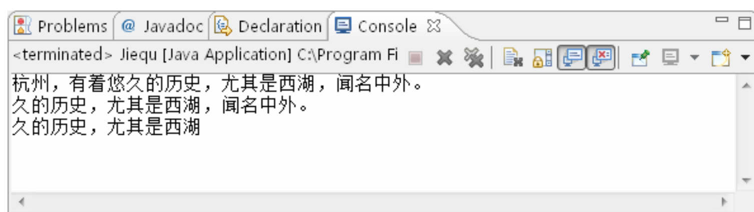


图 4-23　字符串的截取

（7）字符串大小写互转

在许多时候，用户需要对字符串中字母的大小写进行转换，在 String 类里，用户可以使用方法进行互换，将大写字母转换成小写字母可以使用下面的方法，其格式如下：

```
public String toLowerCase（）
```

小写转大写的方法的格式如下：

```
Public String toUpperCase（）
```

下面通过代码将一个字符串里面的大小字母转换成小写字母，其代码见“光盘：源代码/第 4 章/ Daxiao1.java”：

```
public class Daxiao1
 {
   public static void main(String args[])
   {
         String x="I LOVE YoU!!";
      //字母大小写转换
         String y=x.toLowerCase();
         System.out.println(x);
         System.out.println(y);
   }

}
```

运行代码，得到如图 4-24 所示的结果。

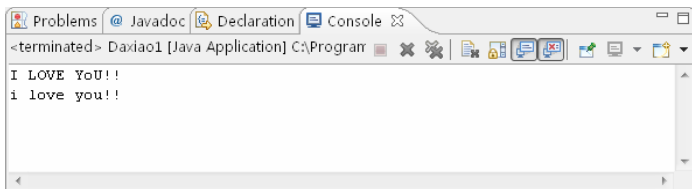
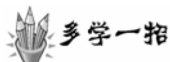


图 4-24　大转小



除了大转小以外，还可以用方法将字符串的小写转换成大写，下面将通过这一方法将已有的小写字符全部转换成大写，其代码见“光盘：源代码/第 4 章/ Daxiao2.java”：

```
public class Daxiao2
{
   public static void main(String args[])
   {
         String x="how are you";
         String y=x.toUpperCase();
         System.out.println(x);
         System.out.println(y);
   }
}
```

运行代码，得到如图 4-25 所示的结果。

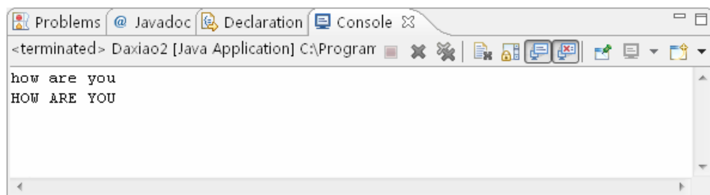


图 4-25　小写转换成大写

（8）消除字符串中的空格字符

在字符串中可能有空白字符，在一些特定的环境中并不需要这样的空白，这时候就去除掉空白，其格式如下：

```
public String trim（）
```

代码 18：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/Daxiao2.java”：

```
public class Kongbai{
   public static void main(String args[]){
         String x=" 想你了，你  在远方还好吗？ ";
         String y=x.trim();
         System.out.println(x);
         System.out.println(y);
   }
}
```

运行代码，得到如图 4-26 所示的结果。

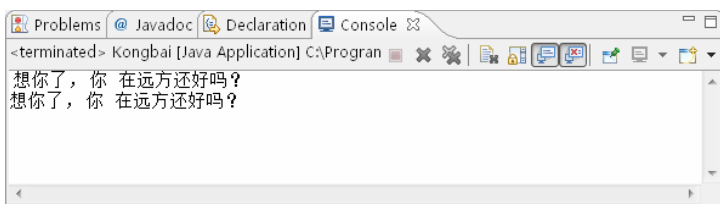


图 4-26　去除空白

提示：使用上面的方法去除空白，只能去除首字母以前的空白，中间的空白不能消除。

4.3.3　StringBuffer 类

StringBuffer 是 Java 中另一种对字符串操作的类，当需要对字符串进行大量的修改时，使用 StringBuffer 类可以大大提高工作效率。在上一节中学习 String 类，这一节中学习字符串 StringBuffer，下面讲解 StringBuffer 中的方法。

（1）追加字符

和 String 一样，在 StringBuffer 类中也可以追加字符，其方法格式如下：

```
public synchronized StringBuffer append
```

代码 19：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/Zhui1.java”：

```
public class Zhui1 {
   public static void main(String args[]){
         StringBuffer x1=new StringBuffer("金山 WPS 办公");
         x1.append("，中国人的选择");
         System.out.println(x1);
         StringBuffer x2=new StringBuffer("WPS");
         x2.append(2009);
```

```
            System.out.println(x2);
        }}
```

运行代码，得到如图 4-27 所示的结果。

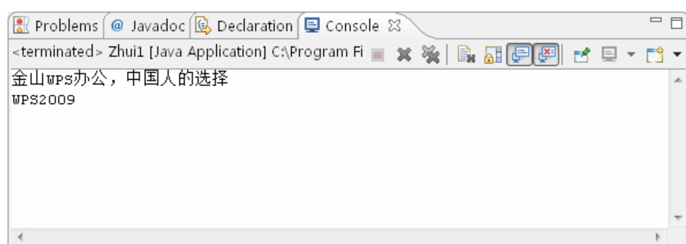


图 4-27　追加字符

（2）插入字符

追加字符只能在末尾添加内容，倘若需要在字符中添加内容，需要用到如下的方法：

```
    public synchronized StringBuffer insert（int offset，String s）
```

参数介绍如下：

这个方法的意思是，将第二个参数的内容，添加到第一个参数指定的位置。换句话说，第一个参数表示要插入的起始位置，第二个参数是需要插入的内容，它可以是包括 String 的任何数据类型。

代码 20：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/Charu.java”：

```
    public class Charu{
        public static void main(String args[]){
            StringBuffer sb1 = new StringBuffer("我的心在跳动。");
            sb1.insert(4,"为你，时时刻刻为你");
            System.out.println(sb1);
        }
    }
```

运行代码，得到如图 4-28 所示的结果。

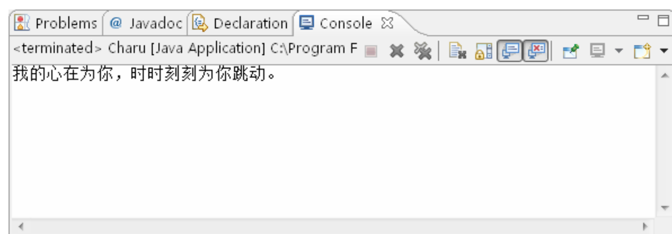


图 4-28　插入字符

（3）颠倒字符

颠倒字符方法主要用于调换字符位置，如“我是谁”，颠倒过来就变成“谁是我”。它的格式如下：

```
public synchronized StringBuffer reverse()
```

代码 21：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 4 章/diandao.java”：

```
public class diandao {
  public static void main(String args[]{
        StringBuffer ch = new StringBuffer("没有什么好说的，请相信我");
        ch.reverse();
        System.out.println(ch);
  }
}
```

运行代码，得到如图 4-29 所示的结果。

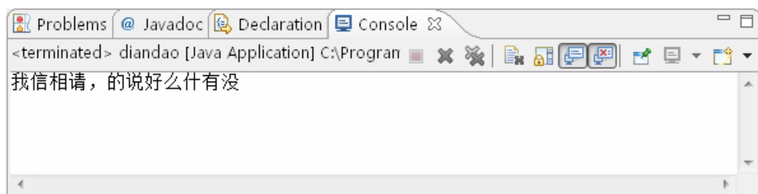


图 4-29　颠倒顺序

4.4　疑难问题解析

本章详细介绍了运算符、表达式、字符串等基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：操作字符串的方法有哪些？在使用时感觉方法不够用，比如转义字符的处理，该怎么解决呢？

解答：操作字符串的方法很多，读者不要误以为字符串的操作就是书中讲解的这几种方法，上面讲解的字符串的操作只是常见的几种方法。读者若需要其他操作字符串的方法，可以查询 Java API 手册。

读者疑问：在开发程序的过程中，常看到只要有表达式，就有运算符，它们是什么关系？

解答：一个 Java 程序，可以没有运算符，但一定有表达式。运算符出现在 Java 程序中，是为了让表达式实现一些计算功能。



职场点拨——提高你的职场生存能力

职场是人生中一个重要的阶段，在人的一生中，大部分时间都会在职场中打拼。对于刚刚进入职场的大学生，可以通过以下 6 个方法强化自己在职场中的生存能力。

（1）沉着冷静

作为职场新人，工作中难免会遇到许多不愉快的事情，但无论如何都要记住，凡事要

“三思而后行”，务必先冷静几分钟，着眼大局。

（2）忠厚为本

对于职场新人，要忠厚老实，切忌偷奸耍滑，人前人后要保持一致，要树立起自己诚信的口碑。

（3）有所担当

对于职场新人，要担当一定的责任，许多职场新人缺乏承担责任的勇气。当在工作中犯下错误时，作为职场新人，不应该掩饰过错，应该和领导或者同事沟通，讨教解决问题的方法。如果只是一味地掩饰错误，被同事或领导发现，结局往往会很惨。

（4）仔细分析

在工作中遇到问题先冷静分析，看清问题的关键所在，假如确实是能力欠缺，不妨提一些建议，也许这些建议会显得稚嫩，但也可能成为激活整个工作进程的动力。

（5）学会包容

在工作中不要过度表现个性，在上司和同事面前一定要树立良好的形象，为人处世学会包容，切记要收敛“个性”和“脾气”。

（6）守口如瓶

在办公室里一定要守口如瓶、心无旁骛。每一个职场人士都有怨言、都需要发泄，但是倒苦水绝对不应该以办公室为对象，因为没有一个领导希望下属动摇军心。

第 5 章　Java 中的假设语句

在 Java 程序中常用假设判断不同条件的结果。本章将带领读者朋友一起领会 Java 中的条件语句，并通过具体实例的实现过程讲解各个知识点的具体使用流程。本章主要内容如下：

- if 语句。
- switch 语句。
- 职场点拨——创业还是就业。

2010 年 X 月 X 日，周六，天气晴。

今天好累啊，刚参加了登山运动！一大早就跟着浩浩荡荡的队伍上山了。山上的岔路真多，经过一路疯狂拔高后，忽然看见前面的队伍停下了，我心中一阵窃喜，以为终于可以休息了。结果上来一看，竟然是迷路了，面前有三条岔路，向导也不知该走哪条路。



一问一答

小菜：“爬山好累啊，中途还迷路了，当时感觉很害怕！”

Wisdom：“那你们怎么走出来的？”

小菜：“向导让我们在岔路口等，他选了一条路走下去，走到一半发现错了，就原路返回，然后继续走另外一条，发现又错了，原路返回，剩下的那条就是正确的了！”

Wisdom：“排查法，你们选择路的过程还真像本章将要学习的‘流程控制语句’！”

小菜：“流程控制语句？”

Wisdom：“对，爬山需要选择正确的道路，而 Java 程序执行时也需要选择将要执行的哪条语句，这个选择过程就需要利用流程控制语句来实现！”

5.1　if 语句

if 语句是假设语句，是最基础的条件语句，本节将会对读者详细讲解关系条件和逻辑条件的 if 语句。

5.1.1　if 控制语句

if 控制语句由保留字符 if、条件语句和位于后面的语句组成。条件语句通常是一个布尔

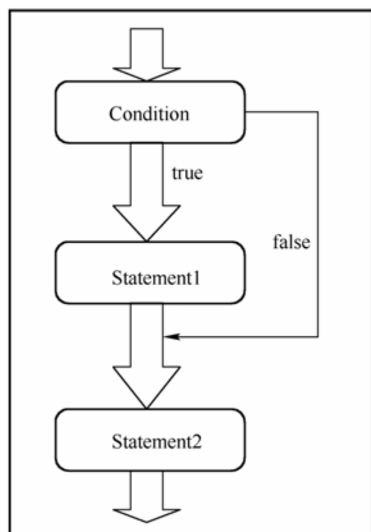


图 5-1　if 语句流程图

表达式，结果为 true 或 false。若条件为 true，则执行语句并继续处理其后的下一条语句，若条件为 false，则跳过该语句并继续处理其后的下一条语句，当条件 condition 为 true 时，执行 statement1，当 condition 为 false 时，则执行 statement2 语句，其流程图如图 5-1 所示。

if 语句的执行方式很简单，它的功能就像开关一样，下面通过一个实例进行讲解。

实例 11：判断是否及格

下面通过代码判断成绩是否及格，其代码见“光盘：源代码/第 5 章/Ifkong.java”：

```
public class If kong
{
   public static void main(String args[])
   {
         int chengji=45;
         if(chengji>60){System.out.println("及格");
         }
         System.out.println("不及格");
   }
```

运行代码，得到如图 5-2 所示的结果。

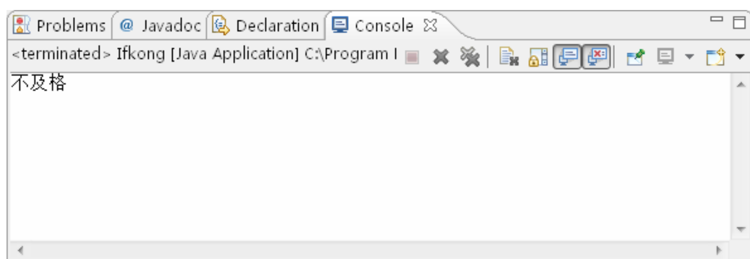
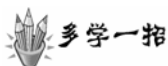


图 5-2　if 语句



在上面的代码中，if 语句的条件不满足，没有执行 if 语句里面的内容，下面展示一个代码，其条件满足的 if 语句，其代码见“光盘：源代码/第 5 章/Ifdan”：

```
public class Ifdan
{
   public static void main(String args[])
   {
         int chengji=100;
         if(chengji>90){System.out.println("优秀");
         }
```

```
            System.out.println("检查完毕");
        }
    }
```

运行代码，得到如图 5-3 所示的结果。

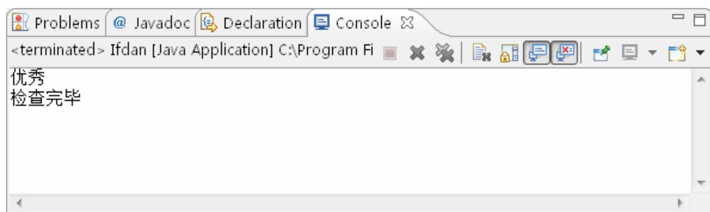


图 5-3　if 语句

5.1.2　if 语句的延伸

在上面的第一个 if 语句中可以看到，它并不会对条件不符合的内容进行处理，这在程序中是不可饶恕的错误，是不允许的。于是就引进了另外一种条件语句，if-else 语句，其基本形式如下：

```
If（condition）statement1;
else statement2;
```

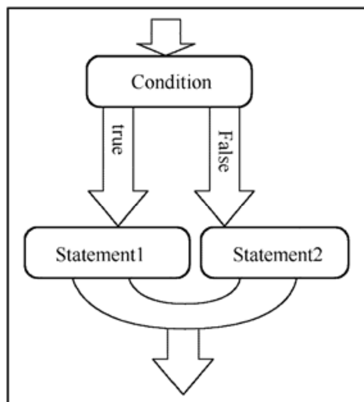


图 5-4　if-else 语句

它的执行流程如图 5-4 所示。

实例 12：if 实现两种条件判断

if-else 可以对两种条件进行判断，并给出不同的答案。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 5 章/Ifjia.java”：

```
public class If jia
 {
   public static void main(String args[])
   {
         int a = 100;
         if(a>99)
               {
               System.out.println("我的值大于 99");
}
         else {
               System.out.println("我的值小于等于 99");
}
         System.out.println("检验完毕");
                     }
}
```

运行代码，得到如图 5-5 所示的结果。

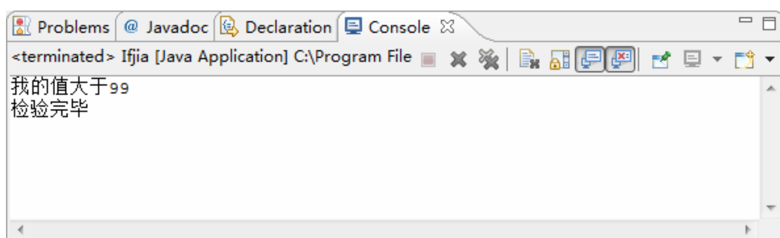
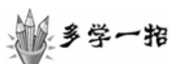


图 5-5　if-else 语句



实际上只有 if-else 的条件语句才能进行真正的判断。在上一节中只存在一种条件，这种程序很少出现，而 if-else 语句针对两种状态，不管是否符合条件，都会给出一个结果。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 5 章/Ifjia2.java”：

```
public class If jia2
{
   public static void main(String args[])
   {
         int 成绩=55;
         if(成绩>60)
               {
         System.out.println("及格了");
         }
         else
{
               System.out.println("没有及格，准备补考");

}
         System.out.println("查询完毕");
         }
}
```

运行代码，得到如图 5-6 所示的结果。

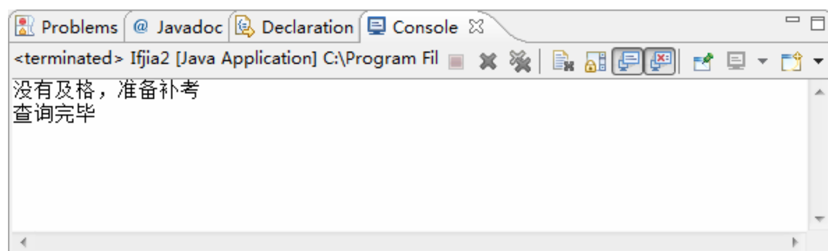


图 5-6　条件语句

提示：在 Java 程序设计里，变量可以是中文。

5.1.3　多个条件判断的 if 语句

if 语句是一种十分强大的条件语句，它可以对多种情况进行判断，在同时判断多个条件时，使用的是 if–else–if 语句，其格式如下：

```
if（condition1）
statement1；
else if（condition2）
statement2；
else statement3
```

参数介绍如下：

首先它会判断第一个条件 condition1 为 True 时，执行 statement。当执行为 false 时，则继续执行下面的代码；当 condition2 为 true 时，执行 statement2；当 condition 为 false 时，则执行 statement3。

if–else–if 的流程图如图 5–7 所示。

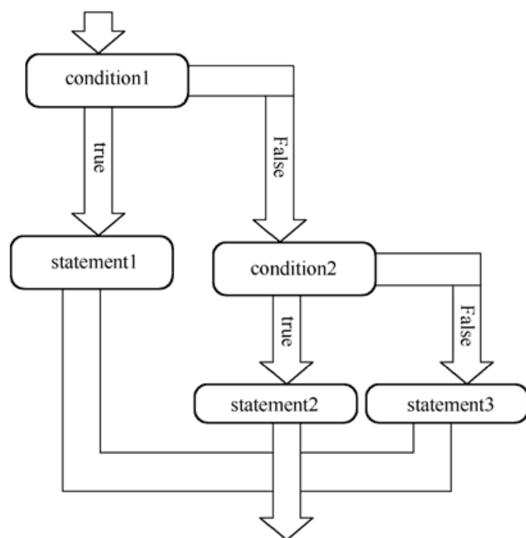


图 5–7　嵌套的 if 语句

实例 13：多条件判断

if–else–if 语句可以对多个条件进行判断，然后给出不同的值，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 5 章/IfDuo.java”：

```
public class IfDuo
 {
          public static void main(String args[])
     {
          int 总成绩=452;
          if(总成绩>610)
              System.out.println("考上重点本科线");
```

```
else if(总成绩>570)
   System.out.println("一般本科线");
else if(总成绩>450)
   System.out.println("专科线");
else if(总成绩>390)
         System.out.println("高职专科线");
else
   System.out.println("没有上线");
System.out.println("检查完毕");
}
```

运行代码，得到如图 5-8 所示的结果。

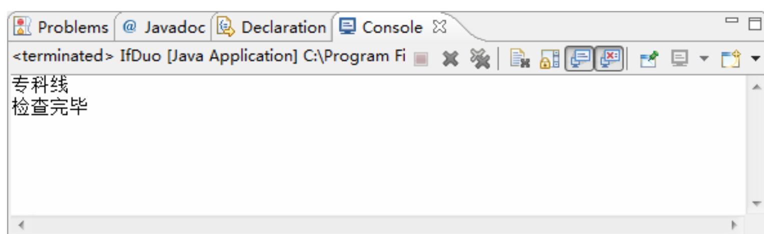


图 5-8　多条件判断

提示：else-if 可以嵌套无限次，只要遇到正确的 condition，就能执行相关的语句，然后结束程序。

多学一招

If-else-if 语句是嵌套式语句，是对多状态进行判断的语句，其实 if 语句可以对一件事物进行多个条件限制。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 5 章/LeapYear.java”：

```
public class LeapYear
{
public static void main(String args[])
{
int year=1989;//method1
if((year%4==0&&year%100!=0)||(year%400==0))
System.out.println(year+"isaleapyear.");
else
System.out.println(year+"isnotaleapyear.");
year=2000;//method2
boolean leap;
if(year%4!=0)
leap=false;
else if(year%100!=0)
leap=true;
else if(year%400!=0)
```

```
leap=false;
else
leap=true;
if(leap==true)
System.out.println(year+"isaleapyear.");
Else
System.out.println(year+"isnotaleapyear.");
year=2050;//method3
if(year%4==0){
if(year%100==0){
if(year%400==0)
leap=true;
else
leap=false;
}else
leap=false;
}else
leap=false;
if(leap==true)
System.out.println(year+" is a leap year.");
else
System.out.println(year+" is not a leap year.");
}
}
```

运行代码，得到如图 5-9 所示的结果。

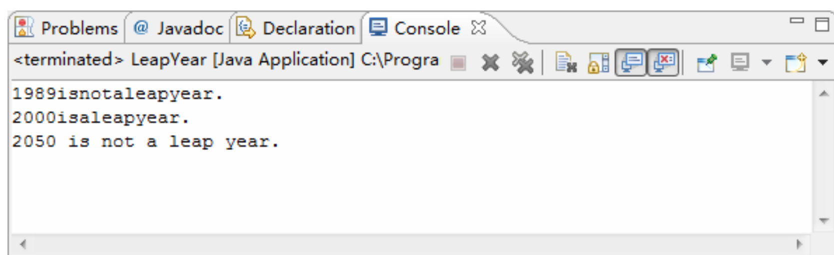


图 5-9　条件语句

5.2　switch 语句

switch 语句方法就是为了判断多条件而诞生的，它的使用和 if 嵌套语句十分相似，但是比它更为简单，下面对其进行详细讲解。

5.2.1　switch 语句的形式

switch 语句是一个对条件进行判断的条件语句，它的格式如下：

```
switch（experssion）
{
Case value1：statement1；
break；
case value2：statement2；
break；
case value3：statement3；
break；
default：statement4；
}
```

它的执行流程如图 5-10 所示。

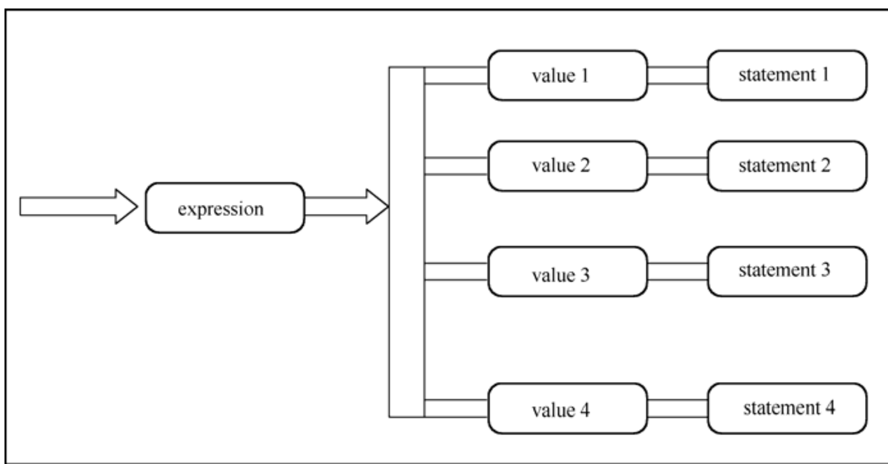


图 5-10　switch 语句

实例 14：使用 switch 语句

表达式 expression 必须是 byte、short、int 和 char 类型，每个 value 必须是与 experssion 类型兼容的一个常量，而且不能重复。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 5 章/switchtest1.java”：

```
public class switchtest1
{
   public static void main(String args[])
   {
         int a=567;
         switch(a)
         {
               case 555:
                  System.out.println("a=555");
                  break;
               case 557:
                  System.out.println("a=557");
                  break;
```

```
                case 567:
                   System.out.println("a=567");
                   break;
                default:
                   System.out.println("no");
          }
   }
}
```

运行代码，得到如图 5-11 所示的结果。

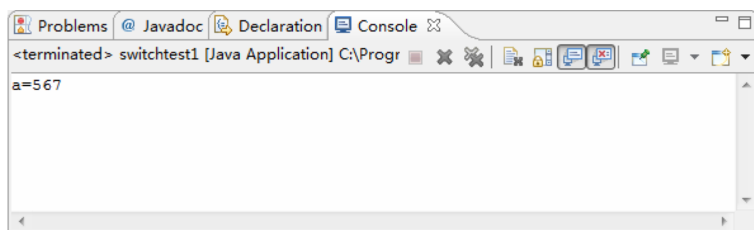
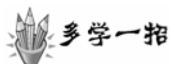


图 5-11　执行结果



上面的代码对条件 switch 语句进行了初步讲解，下面展示另一段代码，其代码见“光盘：源代码/第 5 章/switch2.java”：

```
public class switch2
{
   public static void main(String args[])
   {
          int c=66;
          switch(c)
          {
                case 1:
                   System.out.println("c=1");
                   break;
                case 2:
                   System.out.println("c=2");
                   break;
                case 9:
                   System.out.println("c=9");
                   break;
                default:
                   System.out.println("没有");
          }
   }
}
```

运行代码，得到如图 5-12 所示的结果。

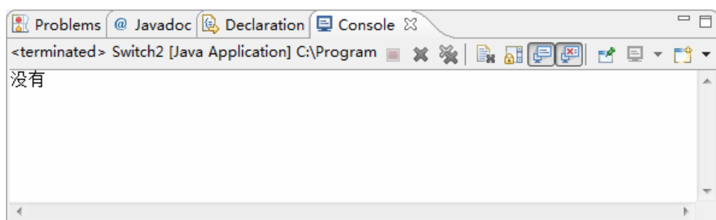


图 5-12　执行结果

5.2.2　switch 语句无 break

在上面的语句中，出现过很多次 break 语句，其实 switch 语句可以去掉这个关键字，下面通过一个实例进行讲解。

实例 15：在 switch 语句中去掉 break

当 switch 语句遇到关键字“break”时，程序会自动结束 switch 语句，而如果把 switch 语句中的 break 关键字去掉了，则程序将自动运行，直到程序结束。下面将讲解两段代码，让读者理解关键字 break 在 switch 中的作用。第一段代码见“光盘：源代码/第 5 章/switchone1.java”：

```
public class switchone1
{
   public static void main(String args[])
   {
         int a=33;
         switch(a)
         {
               case 11:
                  System.out.println("a=11");
               case 22:
                  System.out.println("a=22");
               case 33:
                  System.out.println("a=33");
               break;
               default:
                  System.out.println("no");
         }
   }
}
```

运行代码，得到如图 5-13 所示的结果。

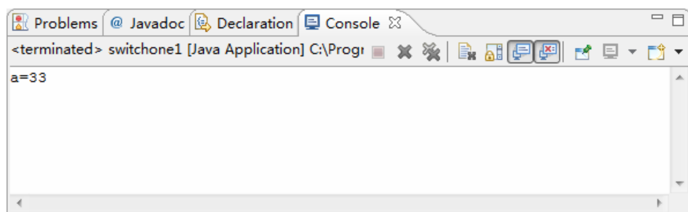
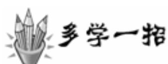


图 5-13　无 break 的 switch 语句



有时在 switch 语句中没有 break 会出现问题，现对上面的代码进行修改，其修改后的代码如下：

```
public class switchone1
{
   public static void main(String args[])
   {
         int a=22;
         switch(a)
         {
               case 11:
                  System.out.println("a=11");
               case 22:
                  System.out.println("a=22");
               case 33:
                  System.out.println("a=33");
               break;
               default:
                  System.out.println("no");
         }
   }
}
```

运行代码，得到如图 5-14 所示的结果。

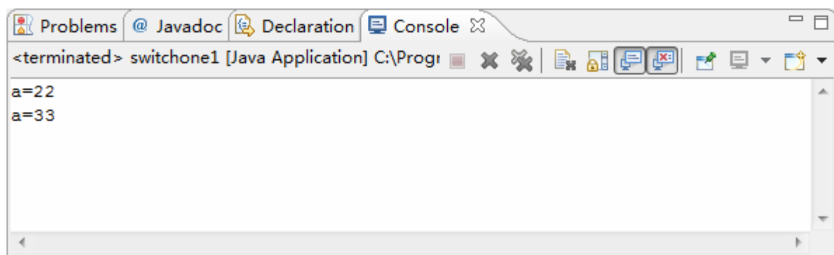


图 5-14　执行结果

提示：从这里大家可以看出 break 的作用，它找到条件符合的内容后仍在继续执行，由此可见 break 语句在 switch 语句中十分重要，如果没有 break 语句，很有可能发生意外。

5.2.3　case 没有执行语句

在前面的讲解中，switch 里的 case 语句都有执行语句，下面通过一个实例进行讲解。

实例 16：case 没有执行语句

在 case 语句后没有执行代码是十分常见的，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 5 章/switchone1.java”：

```
public class switchone1{
   public static void main(String args[])
   {
         int a=111;

         switch(a)
         {
               case 111:
               case 222:
               case 333:
                  System.out.println("a=111|a=222|a=333");
               default:
                  System.out.println("no");
         }}
}
```

运行代码，得到如图 5-15 所示的结果。

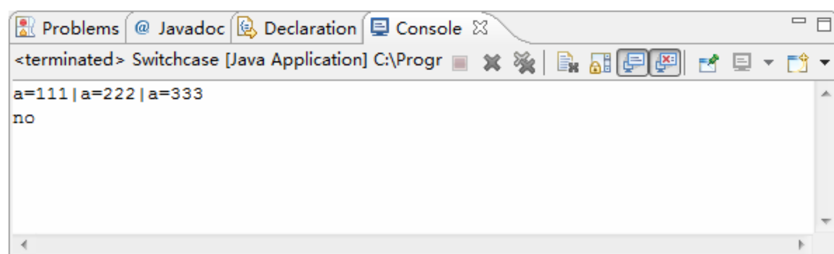


图 5-15　执行的语句

多学一招

上述实例中的程序看起来毫无道理，实际上却意味深长。下面将讲解一段代码，带领读者体会这些程序存在的意义。下面将用它判断月份是属于哪一季度，其代码见“光盘：源代码/第 5 章/switchcase1.java”：

```
public class switchcase1 {
   public static void main(String args[]){
         int month = 6;
         switch(month){
               case 12:
               case 1:
               case 2:
                  System.out.println("冬季");
                  break;
               case 3:
               case 4:
               case 5:
                  System.out.println("春季");
```

```
                break;
            case 6:
            case 7:
            case 8:
                System.out.println("夏季");
                break;
            case 9:
            case 10:
            case 11:
                System.out.println("秋季");
                break;
            default:
                System.out.println("输入错误");
        }
    }
}
```

运行代码，得到如图 5-16 所示的结果。

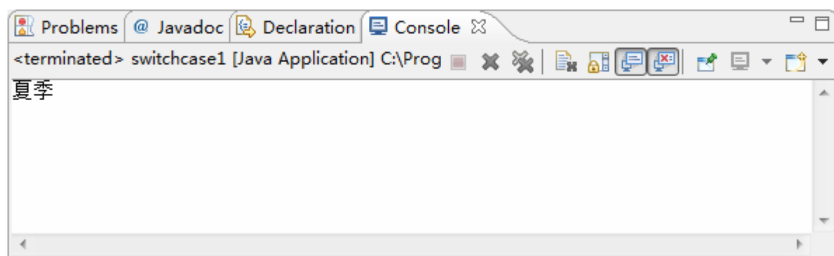


图 5-16　执行结果

5.2.4　default 可以不在末尾

通过前面的学习，许多初学者可能会误认为 default 一定位于 switch 语句的结尾，其实不然，它可以位于 switch 语句的任意位置，请看下面一段代码：

```
public class switch1
{
    public static void main(String args[])
    {
        int a=1997;
        switch(a)
        {
            case 1992:
                System.out.println("a=1992");
            default:
                System.out.println("no");
            case 1997:
                System.out.println("a=1997");
```

```
                    case 2008:
                      System.out.println("a=2008");
                }
            }
        }
```

这段代码很好理解，就是从 a 对应着的语句开始向下执行，直到程序结束。如果下面没有相对应的语句，则从 default 开始执行，直到程序结束。运行代码，得到如图 5-17 所示的结果。

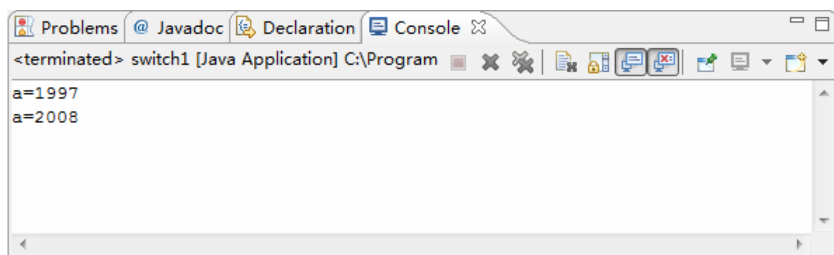


图 5-17　执行结果

5.3　条件语句

至此，用户已经学完 Java 程序设计里的所有的假设语句，用户可以随心所欲编写程序，但是由于初学者编程经验尚浅，缺少处理问题的经验，因此本节将通过深入剖析经典的条件语句，以提升用户的实际编程能力。

5.3.1　正确使用 switch 语句

switch 语句是控制选择的一种方式，编译器生成代码时可以对这种结构进行特定的优化，从而产生效率比较高的代码。在 java 中，编译器根据分支的情况，分别产生 tableswitch 和 lookupswitch 两种，其中 tableswitch 适用于分支比较集中的情况，而 lookupswitch 适用于分支比较稀疏的情况。

代码 22：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 5 章/Testone1.java”：

```
public class Testone1
{
    public static void main(String[] args)
{
          int i=3;
          switch (i)
{
              case 0:
                  System.out.println("0");
```

```
                    break;
                case 1:
                    System.out.println("1");
                    break;
                case 3:
                    System.out.println("3");
                    break;
                case 5:
                    System.out.println("5");
                    break;
                case 10:
                    System.out.println("10");
                    break;
                case 13:
                    System.out.println("13");
                    break;
                case 14:
                    System.out.println("14");
                    break;
            default:
            System.out.println("default");
                    break;
            }
              }
    }
```

上面的 switch 语句是简单的代码，在编写代码时用户一定要清楚，当 case 参数和 switch 参数的值相等时，系统就会执行对应的 case 语句。同时，在 Java 中规定，case 参数必须是常量表达式，也就是 case 语句参数必须是最终的，即 case 的值只能使用常量的最终变量。运行代码，得到如图 5-18 所示的效果。

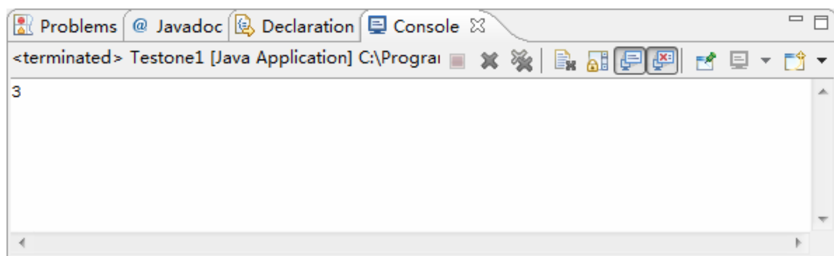


图 5-18　正确编写 switch 语句

5.3.2　正确使用 if 语句

条件语句是程序中根据条件是否成立进行选择执行的一类语句，这类语句在实际使用中，难点在于如何准确地抽象条件。例如实现程序登录功能时，如果用户名和密码正确，则进入系统，否则弹出“密码错误”这样的提示框等。

代码 23： 下面通过一段经典代码进行讲解，其代码见“光盘：源代码/第 5 章/Ifjing.java”：

```
public class Ifjing
{
   public static void main(String[] args) {
          int month=3;
             int days=0;   //日期数
             if(month==1){
             days=31;
             }else if(month==2){
             days=28;
             } else if(month==3){
             days=31;
             } else if(month==4){
             days=30;
             } else if(month==5){
             days=31;
             } else if(month==6){
             days=30;
             } else if(month==7){
             days=31;
             } else if(month==8){
             days=31;
             } else if(month==9){
             days=30;
             } else if(month==10){
             days=31;
             } else if(month==11){
             days=30;
             } else if(month==12){
             days=31;
             }
             System.out.print(days);
               }
}
```

每个 else if 语句在书写时都是有顺序的，在实际书写过程中，必须按照逻辑上的顺序进行书写，否则将出现逻辑错误。上面这段代码的运行结果如图 5-19 所示。

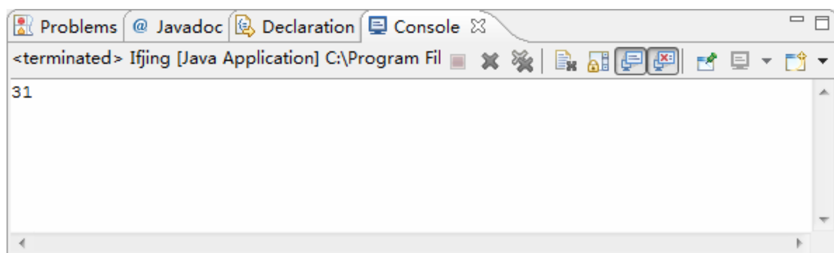


图 5-19 if 语句的正确书写

5.3.3　switch 语句的执行顺序

很多读者在学完 switch 语句后依然不知道 switch 语句的执行顺序是怎么一回事，从前面的知识可以知道，是根据 switch 表达式的值决定选择哪个 case 分支，如果找不到相应的分支，就直接从"default" 开始输出。当程序执行一条 case 语句后，因为例子中的 case 分支中没有 break 和 return 语句，所以程序会执行紧接于其后的语句，

代码 24：下面通过三段代码进行讲解，第一段代码见“光盘：源代码/第 5 章/switchs1.java”：

```
public class switchs1 {
   public static void main(String[] args){
   int x=0;
  switch(x){
 default:
   System.out.println("default");
 case 1:
       System.out.println(1);
     case 2:
          System.out.println(2);
   }
 }
}
```

运行代码，得到的结果是 default 1 2。

第二段代码，代码如下：

```
public class switchs2 {
    public static void main(String[] args) {
                 int x = 0;
                  switch (x) {
                   default:
                     System.out.println("default");
                  case 0:
                      System.out.println(0);
                 case 1:
               System.out.println(1);
            case 2:
                      System.out.println(2);
                  }
}
}
```

运行代码，得到的结果是 0 1 2。

第三段代码，代码如下：

```
public class switchs3 {
```

```
        public static void main(String[] args) {
                    int x = 0;
                     switch (x) {
                     case 0:
                         System.out.println(0);
                     case 1:
                         System.out.println(1);
                   case 2:
                        System.out.println(2);
                    default:
                       System.out.println("default");
                     }
      }
      }
```

运行代码，得到的结果是 0 1 2 default。

5.4 疑难问题解析

本章详细介绍了 if 语句、switch 语句等基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：switch 语句和 if 语句都具有嵌套功能，都可以通过嵌套对多个条件进行判断，在编程过程中使用哪个语句比较好？

解答：这要视具体情况而定，采用 if-else-if 语句格式实现多分支结构，实际上是将问题细化成多个层次，并对每个层次使用单、双分支结构的嵌套。而 switch 语句则可以直接处理多分支选择结构，当某种算法要用某个变量或表达式单独测试每一个可能的整数值常量，然后做出相应的动作时，则 switch 语句比 if 优秀得多。

读者疑问：switch 语句有着很强大的功能，但它的执行顺序是怎样的呢？

解答：switch 语句的执行顺序是选择性执行，首先对其条件进行判断，如果条件符合，则会执行对应的语句块。



职场点拨——创业还是就业

大学生，有学计算机的，有学网站设计的……当他们怀着对社会的美好憧憬走出校园时，都想在社会上创造出自己的一片天地。还有一部分人在校期间参与了多项兼职工作甚至全职工作，积累了一定的职场经验，想在毕业后自己创业。

当前大学生的就业压力很大，因为对自己当前的工作不满意，所以产生了自主创业的念头。其实作为一名刚毕业的学生，最好不要选择创业这条道路，因为缺乏实际经验，又好高骛远，很难成功。笔者的建议是先找个工作，磨炼一下自己，积累一些社会知识和工作经验。通过观察分析他人的成功案例和失败案例后，再结合自己的想法，研究自己的梦想能否

行得通，自己的策划可能会出现哪些问题。

实践证明，在社会上多闯荡几年，积累一定的社会知识后再选择创业，是最理想的成功之道。例如你想做 IT 硬件，你可以进入一家类似的公司，了解他们的运作过程和客户来源。这样既可以熟悉别人的经营模式，也可以实现资金积累，同时还可以学会怎样发现客户和保住客户，毕竟客户才是你的经济来源。

所以，创业的前提是要从所创行业中吸取更多经验，而经验则主要来自产品渠道、客户开发和经营手段。如果你拥有了这些财富，将会降低创业的风险，提高创业的成功率，为自己多买一份“保险”。

第 6 章　循 环 语 句

上一章学习了条件语句，让程序的执行顺序发生了变化。在本章中，为了实现循环和跳转等功能，将为读者详细讲解 Java 编程中重要的语句循环：for 语句、while 语句、do…while 语句以及跳转语句。本章主要内容如下：

- 循环语句。
- 跳转语句。
- 职场点拨——一份简历引发的深思。

2010 年 XX 月 X 日，天气阴。

今天 HR 的同事 A 来倒苦水，说:“公司半年招聘一次，原来招的员工已离职大半。目前的招聘要求很低，但就是招不到合适的。”然后 A 给我看了他们公司的招聘信息:

职位: AA 公司招聘软件工程师

要求如下:

计算机相关专业，专科以上学历，两年以上工作经验。

熟练运用 Java 语言，熟悉编程语言，例如 VB、C 等优先。

熟练使用 SQL Server 2005 数据库，熟悉 MySQL 者优先。

熟悉 Windows 开发环境，熟悉 Linux 者优先。

工作认真细心，能吃苦耐劳，有上进心，有团队合作精神和良好的沟通能力。

6.1　Java 循环语句

在 Java 程序设计中主要有三种循环语句：for 循环、while 语句和 do…while 语句。下面将分别对这三个语句进行详细讲解。

6.1.1　for 循环语句

在 Java 程序设计中，for 语句是最为常见的一种循环语句，它是一个功能强大且形式灵活的结构，下面对它进行讲解。

（1）for 语句的书写格式

for 语句是十分常见的循环语句，它的书写格式如下：

```
for（initialization;condition;iteration）
{
}
```

从上面的代码格式可以看出，for 循环语句是由变量的声明和初始化、布尔表达式、循环表达式三部分组成的，每一部分都用分号分隔。在 for 循环的执行过程中，当循环启动后，最先开始执行的是初始化部分（求解表达式 1），紧接着执行布尔表达式（表达式 2）的值，如果符合条件，则执行循环，不符合条件，则跳出循环。分别介绍如下：

- **声明和初始化**：for 语句中的第一部分是关键字 for 之后的括号内的声明和初始化变量，声明和初始化发生在 for 循环内任何操作前，声明和初始化只在循环时执行一次。
- **条件表达式**：第二部分是条件表达式，它的计算结果必须是布尔值，在 for 循环中，只能有一个表达式。
- **循环表达式**：for 循环体每执行一次后，都会执行循环表达式。它永远在循环体运行后执行，也就是最后执行。

代码 25：下面通过一小段代码体验，for 语句的书写形式，其代码见“光盘：源代码/第 6 章/Forone.java”：

```
public class Forone1 {
   public static void main(String args[])
   {
         for(int a=0;a<13;a++)
         {
               System.out.println(a);
         }
         }
}
```

运行代码，得到如图 6-1 所示的结果。

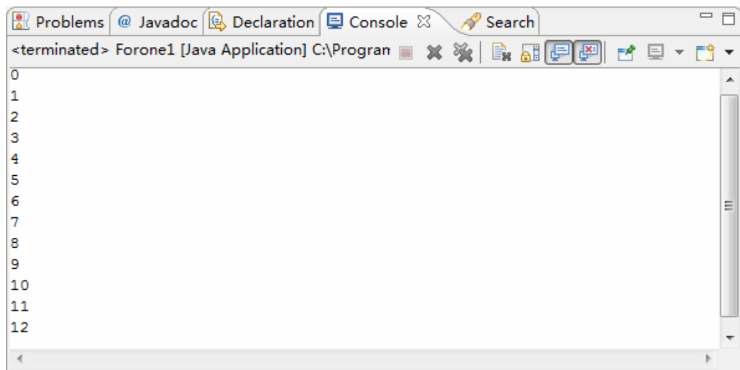


图 6-1　循环语句

（2）for 语句的执行方式

从前面的知识了解了 for 语句的书写格式，那么它是如何执行的呢？如图 6-2 所示为 for 循环执行的流程图。

（3）for 语句的初体验

前面的知识中，用户已经清楚了 for 语句的书写格式和执行方式，下面通过一个实例进

行讲解，它的应用。

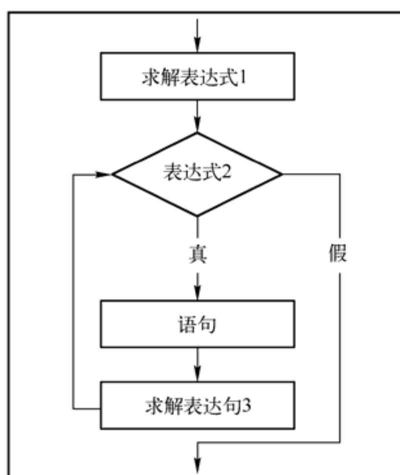


图 6-2　for 循环执行流程图

实例 17：使用 for 语句

for 语句是最常用的循环语句，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 6 章/fortwo1.java”：

```
public class fortwo1
 {
   public static void main(String args[])
   {
//for 循环语句
         for(int j=0;j<8;j++)
         {
               System.out.println("☆");
         }
         }
}
```

运行代码，得到如图 6-3 所示的结果。

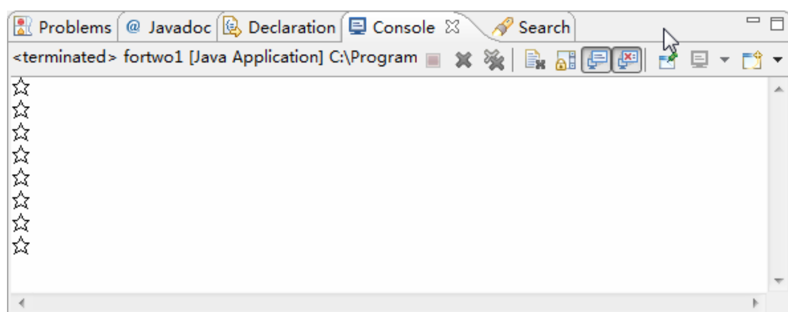
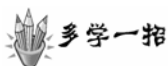


图 6-3　for 语句



For 语句的条件表达式一般情况下是变量，但允许有多个表达式。初始化部分和循环表达式部分声明多个变量，每个变量间用逗号隔开。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 6 章/fortwo2.java”：

```
public class fortwo2
{
   public static void main(String args[])
   {
//for 语句
         for(int Aa=2,Bb=12;Aa<Bb;Aa++,Bb--)
         {
               System.out.println("Aa="+Aa);
               System.out.println("Bb="+Bb);
         }
         }
}
```

运行代码，得到如图 6-4 所示的结果。

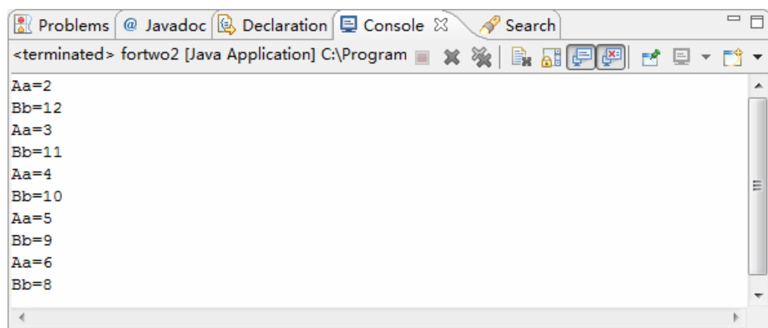


图 6-4　执行结果

（4）for 语句尽显不凡魅力

代码 26：通过前面的学习，用户只是认识了 for 语句，下面通过一段代码展示 for 语句的功能，其代码见“光盘：源代码/第 6 章/fortwo3.java”：

```
public class fortwo3
 {
   public static void main(String[] args)
   {
         //第一层 for 嵌套语句
         for(int a=0;a<3;a++)
         {
               //第二层 for 嵌套语句
               for(int b=a;b<3;b++)
               {
```

```
                    System.out.println("$");
                }
                System.out.print("￥");
            }
        }
    }
```

运行代码，得到如图 6-5 所示的结果。

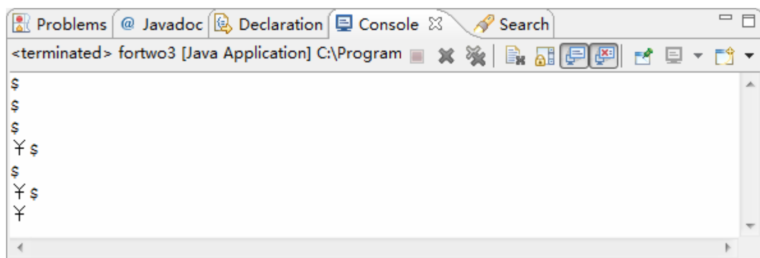


图 6-5　for 循环

从上面的代码可以看出，for 语句中可以植入 for 语句，在 Java 程序设计中，这种现象被称做 for 语句嵌套。for 语句嵌套的最大特点是外层 for 语句执行一次，内层 for 语句执行多次，如果外层 for 语句执行第二次，内层循环又会执行多次。直到外层 for 语句不满足条件时，内层 for 语句才会停止执行。下面通过一个实例讲解 for 语句嵌套的功能和思想。

提示： for 嵌套的形式是这样的：for（m）{for（n）{}}。它执行的方式是 m 循环执行一次，内循环执行 N 次，然后外循环执行第 2 次，内循环再执行 N 次，直到外循环执行完为止，内循环也会终止。

在上面的代码中，读者领悟到了 for 循环语句的嵌套，下面通过一个实例进行讲解。

实例 18：使用 for 语句的嵌套

for 语句的嵌套在 Java 程序中应用得十分普遍，如用“*”型摆放一个星型菱形，就是很经典的程序。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 6 章/fortwo4.java”：

```
public class fortwo4
{
   public static void main(String[] args)
   {
   int n=7;
   int o=(n/2);
   int t=1;
   int step=2;
   for (int i=0; i<n; i++)
   {
   for (int j=0; j<Math.abs(o); j++)
   {
```

```
            System.out.print((char)32);
            }
            o--;
            for (int k=1; k<=t; k++)
            {
            System.out.print("*");
            }
            t=t + step;
            if (t=n)
            {
            step=-step;
            }
            System.out.println();
            }
            }
        }
```

运行代码，得到如图 6-6 所示的结果。

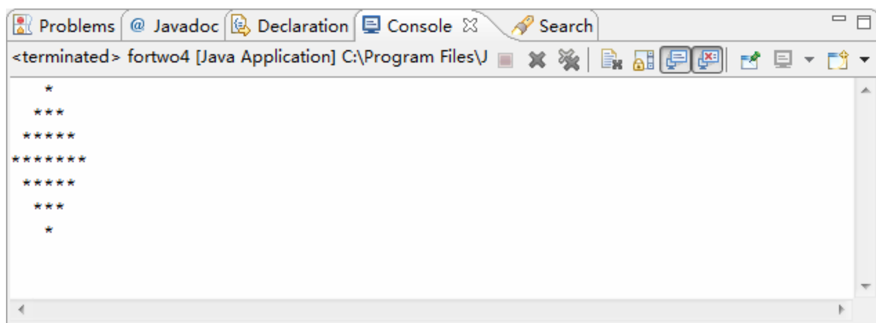


图 6-6　运行的结果

多学一招

For 循环嵌套有着许多经典的程序，除了菱形，还有九九菱形、倒置九九乘法表等，都是利用经典的循环语句完成的。下面通过 for 循环嵌套语句编写一个倒置的三角图，其代码见“光盘：源代码/第 6 章/fortwo4.java”：

```
        public class fortwo5
         {
           public static void main(String[] args)
            {
             for(int a=13; a>=1; a--)
             {
              for(int b=a; b>=1; b--)
              {
              System.out.print("☆" + " ");
               }
```

```
            System.out.println();
        }
      }
}
```

运行代码，得到如图 6-7 所示的结果。

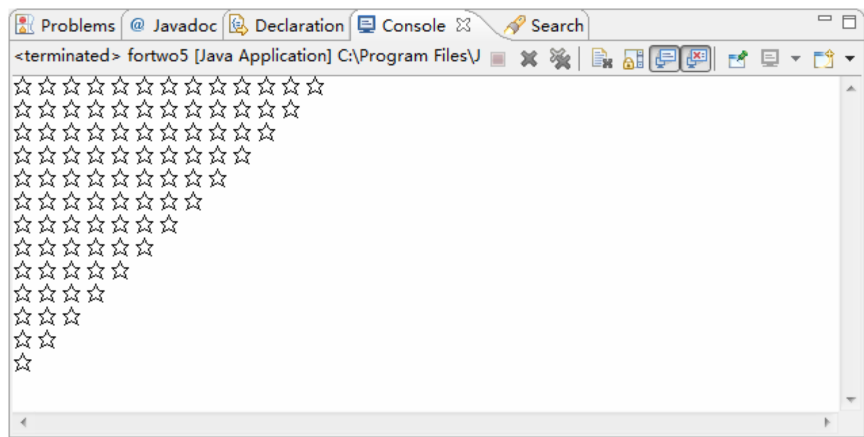


图 6-7　for 语句的嵌套

提示：循环嵌套应用十分广泛，读者朋友应多思考，把知识融会贯通，并尝试独立编写一个程序，以得到如图 6-8 所示的效果。

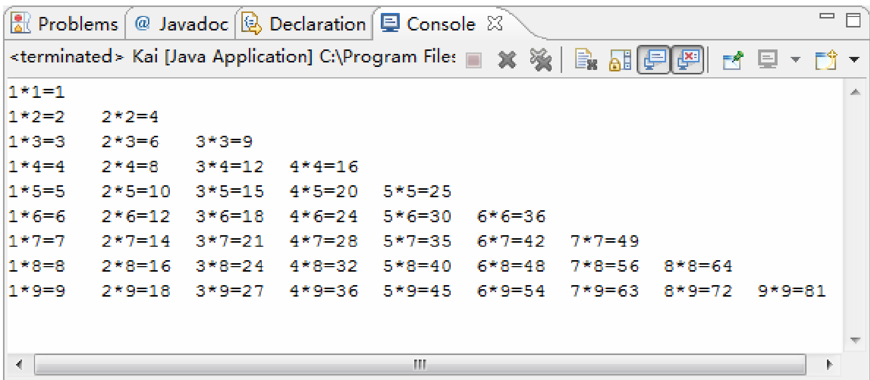


图 6-8　九九乘法表

6.1.2　while 循环语句

在 Java 程序中除了 for 循环语句以外，while 语句也是十分著名的循环语句，它的特点和 for 语句十分类似，但并非完全相同。下面对它进行讲解。

while 循环语句最大的特点就是不知道会循环多少次。当不知道语句块或语句需要重复多少次时，使用 while 语句无疑是最好的选择。当它的表达式是真时，while 语句重复执行一条语句或者语句块。它的基本格式如下：

```
While（condition）
{
}
```

它的执行流程图也十分简单，如图 6-9 所示。

代码 27：下面通过一段简单的代码认识 while 循环语句。其代码见“光盘：源代码/第 6 章/whileone.java”如下：

```
public class whileone
{
   public static void main(String args[])
   {
         int X=0;
//while 循环语句
while(X<19)
         {
               System.out.print(X);
               X++;
         }
   }
}
```

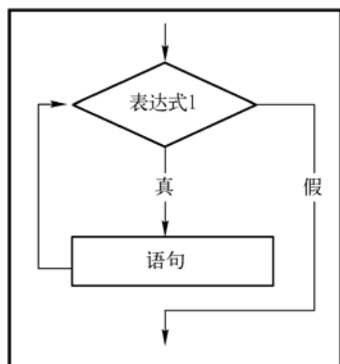


图 6-9　while 语句的执行流程图

在上面的代码中用到了一个简单的 do…while 语句。运行代码，得到如图 6-10 所示的结果。

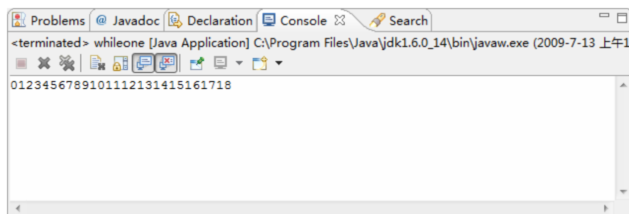


图 6-10　程序执行结果

从上面这个例子中可以看出 while 语句和 for 循环的语句在结构上有很大不同，下面通过一个实例进行讲解。

实例 19：使用 while 循环语句

使用 while 循环语句判断累加和不大 30 的所有自然数，其代码见“光盘：源代码/第 6 章/whiletwo.java”：

```
public class whiletwo
{
   public static void main(String[] args)
   {
         int sum=0;
         int a=1;      //由于是计算自然数，所以 1 的初始值设置为 1
         System.out.println("累加和不大于 30 的所有自然数如下：");
         while(sum<30){
```

```
                    sum=sum+a;
                    System.out.println(a);
                    a++;        //该语句一定不用少
               }
          }
     }
```

运行代码，得到如图 6-11 所示的结果。

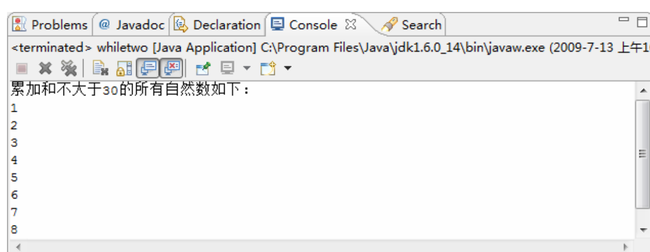
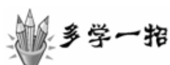


图 6-11　累加的和



while 语句除了上面这种写法外，还有另外一种写法。下面通过一段代码进行讲解，这段代码同时用到了 while 语句和 if 语句，其代码见“光盘：源代码/第 6 章/whilethree.java”：

```
     public class whilethree
     {
        public static void main(String[] args)
        {
     //while 循环语句
        int x=0;
        while(++x<=78)
        if ((x%7)==0)
        System.out.print(x+"\t");
        System.out.println();
        }
     }
```

运行代码，得到如图 6-12 所示的结果。

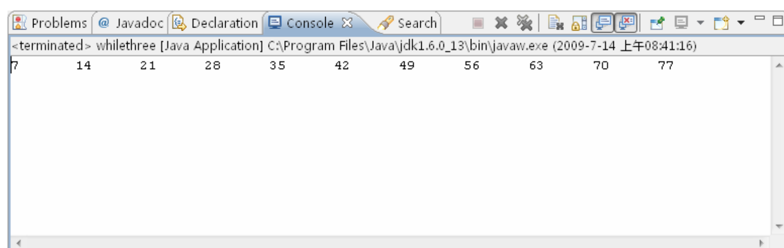


图 6-12　运行结果

6.1.3　do…while 循环语句

在 Java 程序设计里，除了 for 语句实现循环功能外，还有 do…while 语句，do…while 语句最大的特点是先执行表达式，再判断条件，如果条件符合，再执行表达式。下面将详细讲解 do…while 语句。

（1）do…while 的书写格式

Java 为用户提供了这么一种循环的语句，那就是 do…while 循环语句，do…while 语句的特点是至少会执行一次循环体，因为它的条件表达式在循环的最后。do…while 的格式如下：

```
do
{
}
While（condition）
```

do…while 语句是先执行一次，再判断表达式，如果表达式为真，则循环继续，如果表达式为假，则循环到此结束。

代码 28：例如下面一段代码，其代码见“光盘：源代码/第 6 章/doone.java”：

```
public class doone
{
   public static void main(String args[])
   {
         int x=0;
         do
         {
               System.out.println(x);
               x++;
         }while(x<8);
   }
}
```

运行代码，得到如图 6-13 所示的结果。

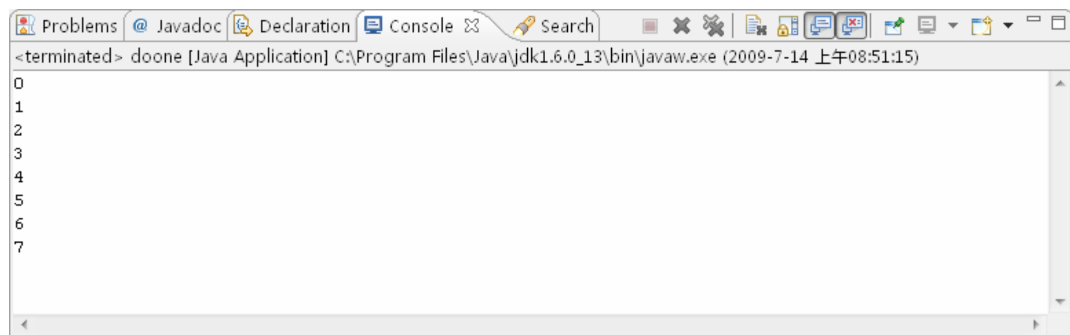


图 6-13　do…while 语句

（2）do…while 的执行方式

通过上面的代码，用户对 do…while 语句的执行方式有了初步的了解，它执行的流程如图 6-14 所示。

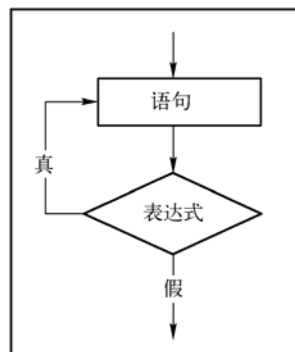


图 6-14 do…while 流程图

前面一直在重复的一个问题就是无论如何 do…while 语句都要执行一次代码。

代码 29： 下面通过一段代码进行讲解，其代码见“光盘：源代码/第 6 章/dotwo.java”：

执行程序，得到如图 6-15 所示的结果。

（3）do…while 的应用举例

do…while 是常见的循环语句，使用它的频率十分之高，下面通过一个实例对它进行学习与理解。

```
public class dotwo
{
   public static void main(String args[])
   {
         int k=5;
         do
         {
               System.out.println(k);
               k++;
         }while(k<-2);
   }
}
```

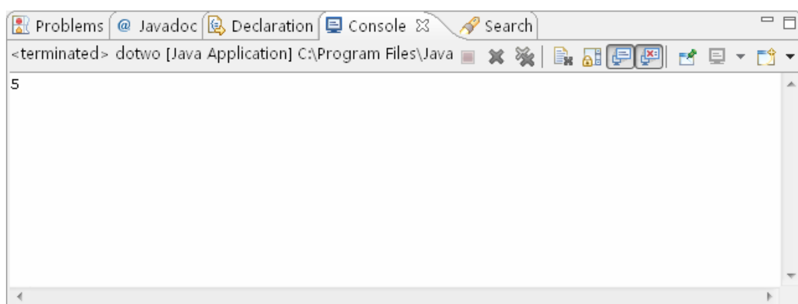


图 6-15 执行结果

实例 20：使用 do…while 循环语句

使用 do…while 循环语句判断累加和不大于 120 的所有自然数，其代码见“光盘：源代码/第 6 章/dothree.java”：

```
public class dothree
{
   public static void main(String args[])
   {
         int i=1;
```

```
            int sum=0;
            do
            {
            sum+=i++;
            }
            while(i<=120);
            System.out.println(sum);
        }
    }
```

提示：在书写 do…while 程序时，千万不要忘记 while ( ) 后面的 “;”，初学者容易漏掉这个分号，这样会导致编译和运行时会报错。

运行代码，得到如图 6-16 所示的结果。

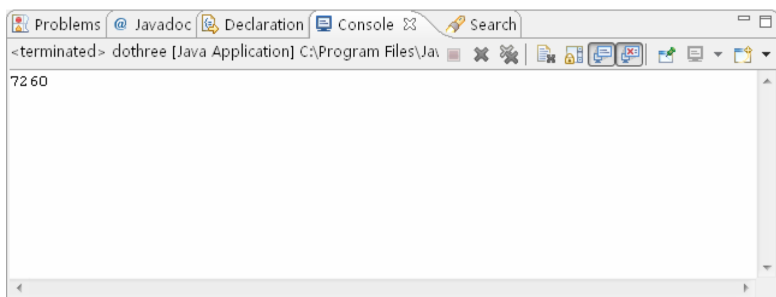


图 6-16　执行结果

多学一招

do…while 除了可以累加的功能外，还有许多功能，它的功能与前面两种循环语句差不多，只有一些微小的区别。下面通过一段代码进行讲解，其代码见 “光盘：源代码/第 6 章/dofour.java”：

```
public class dofour
{
    public static void main(String[] args)
    {
    int ting=25;
    do
    {
    ting=ting-5;
    System.out.println(ting);
    }
    while (ting>=5);
    }
}
```

运行代码，得到如图 6-17 所示的结果。

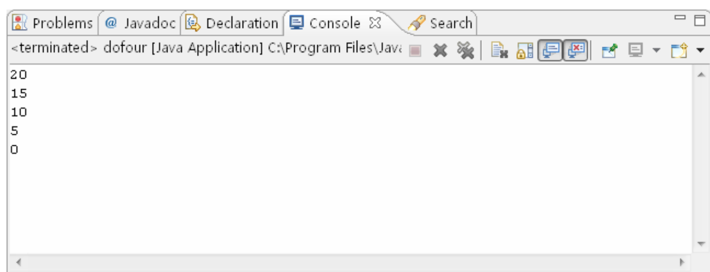


图 6-17　执行结果

6.2　跳转功能的实现

在条件语句和循环语句中若不需要再进行循环，就需要使用特定的关键字，如 break 关键字。除了 break 关键字外，还有许多关键字可以实现跳转功能，本节将详细讲解跳转语句。

6.2.1　break 语句的应用

在前面，用户已经接触过了 break 语句，它可以在 switch 语句里终止一个语句。除了这个功能外，它还具有其他的功能，比如可以用来退出一个循环。break 语句根据用户的使用不同，可以分为无标号退出循环和有标号退出循环，下面将对它们进行讲解。

（1）无标号退出循环

无标号退出循环就是直接退出循环，在循环语句中遇到 break 语句，循环就会立即终止，循环体外面的语句也将会重新开始。

代码 30：下面将通过一段代码，对无标号退出循环语句进行简单的认识，其代码见“光盘：源代码/第 6 章/break1.java”：

```
public class break1
{
   public static void main(String args[])
   {
         for(int dd=0;dd<19;dd++)
         {
               if(dd==3)
               {
               //跳转功能从此开始
                     break;
               }
               System.out.println(dd);
         }
   }
}
```

从上面的程序可以看出，不管 for 循环会循环多少次，它都会在“d=3”时终止程序。

运行代码，得到如图 6-18 所示的结果

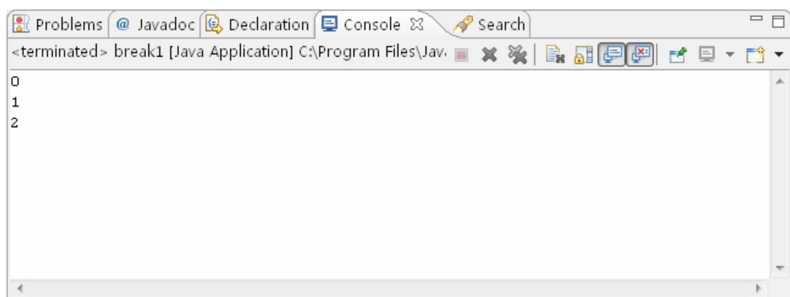


图 6-18　无标号退出循环

break 语句不但可以用于 for 语句还可以用于 while 语句和 do…while 语句里，下面通过一个实例对它们进行讲解。

实例 21：在 while 循环语句中使用 break

在 while 循环语句中使用 break 关键字，其代码见“光盘：源代码/第 6 章/break2.java”：

```
public class break2
 {
   public static void main(String args[])
   {
         int A=0;
         while(A<18)
         {
               if(A==7)
               {
                     break;
               }
               System.out.println(A);
               A++;
         }
   }
}
```

运行代码，得到如图 6-19 所示的结果。

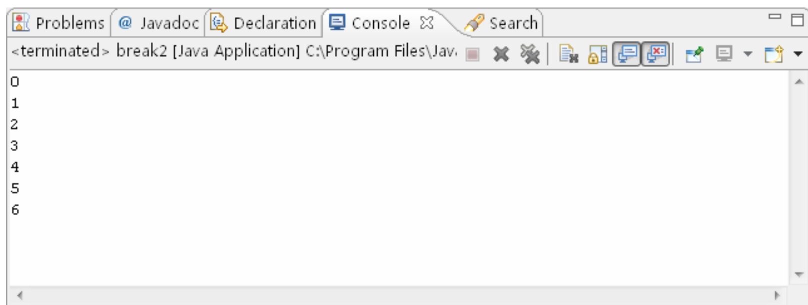
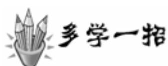


图 6-19　执行代码



上面的代码讲解了在 while 语句中使用 break 语句的方法，下面再通过一段代码讲解在 do…while 语句中使用 break 语句的方法，其代码见“光盘：源代码/第 6 章/break3.java”：

```
public class break3
{

   public static void main(String args[])
   {
         int A=0;
         do
         {
               if(A==5)
               {
                     break;
               }
               System.out.println(A);
               A++;
         }
         while(A<9);
   }
}
```

运行代码，得到如图 6-20 所示的结果。

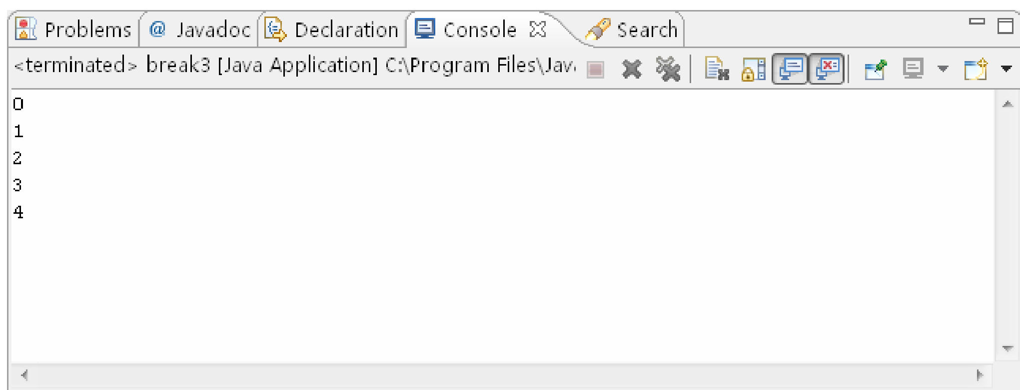


图 6-20　执行代码

（2）无标号退出循环的应用

在上一节里，用户对无标号退出循环有了初步理解，下面通过一个实例讲解它是如何应用在实际的程序设计里。

实例 22：在嵌套语句中使用 break 语句

break 语句除了上述用法外，还可以用在嵌套语句中，它将终止它所在的循环。接下来先让读者朋友看一下终止内层循环的程序，其代码见“光盘：源代码/第 6 章

/breakqian1.java”：

```
public class breakqian1
{
   public static void main(String args[])
   {
         for(int i=1;i<12;i++)
         {
               for(int j=i;j<12;j++)
               {
                     if(j==6)
                     {
                           break;
                     }
                     System.out.print("◇");
               }
               System.out.println();
         }
   }
}
```

运行代码，得到如图 6-21 所示的结果。

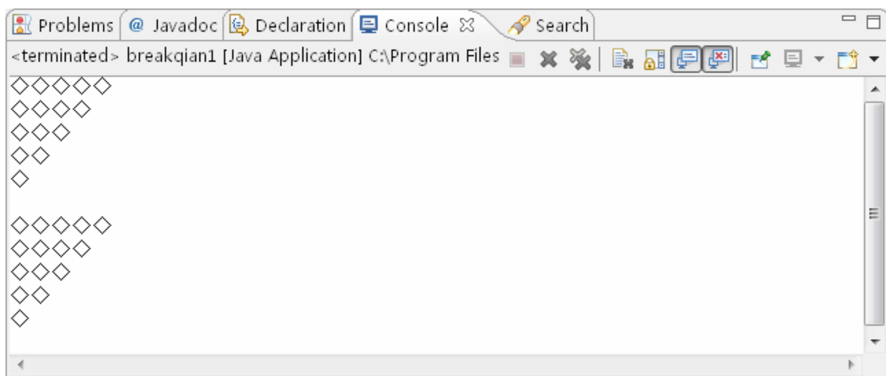


图 6-21　嵌套语句中的 break

多学一招

break 语句除了上述用法外，还可以用在嵌套语句中的外层。上面实例是将 break 语句用在嵌套语句的内层，下面将展示一段代码，讲解在循环语句的外层使用 break 语句的方法，其代码见“光盘：源代码/第 6 章/breakqian2.java”：

```
public class breakqian2
{
   public static void main(String args[])
   {
         for(int i=1;i<12;i++)
```

```
            {
                if(i==7)
                {
                    break;
                }
                for(int j=i;j<10;j++)
                {
                    System.out.print("※");
                }
                System.out.println();
            }
        }
    }
```

运行代码，得到如图 6-22 所示的结果。

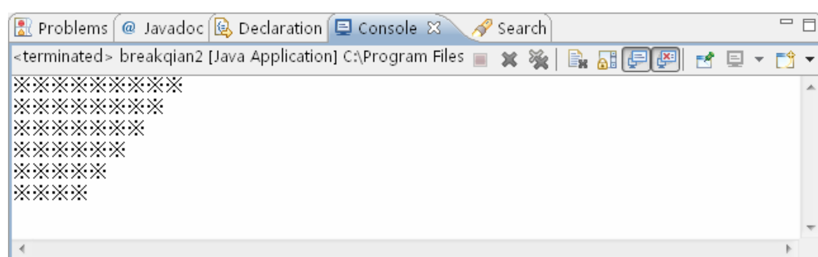


图 6-22　嵌套语句的 break

（3）有标号的 break 语句

只有在嵌套的语句中，才可以使用有标号的 break 语句。在嵌套的循环语句中，可以在循环语句前加一个标号，这样，在使用 break 语句时，就可以使用 break 后面紧接着的循环语句前的标号来退出该标号所在的循环了，下面通过一个实例进行讲解。

实例 23：使用有标号语句

有标号语句只能用在嵌套的循环语句中，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 6 章/breakyou.java”：

```
public class breakyou
 {
   public static void main(String args[])
   {
         out:for(int X=0;X<10;X++)
             {
               System.out.println("X="+X);
               for(int Y=0;Y<10;Y++)
               {
                     if(Y==7)
                     {
                           break out;
```

```
                        }
                        System.out.println("Y="+Y);
                    }
                }
        }
    }
```

运行代码，得到如图 6-23 所示的结果。

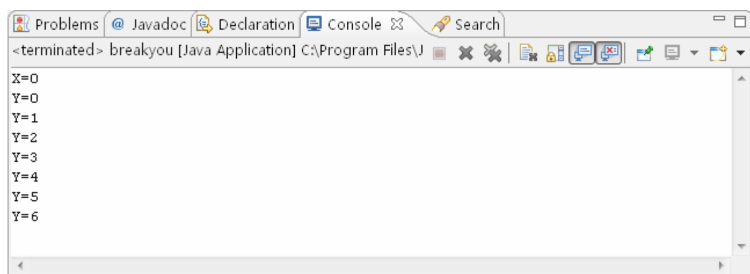


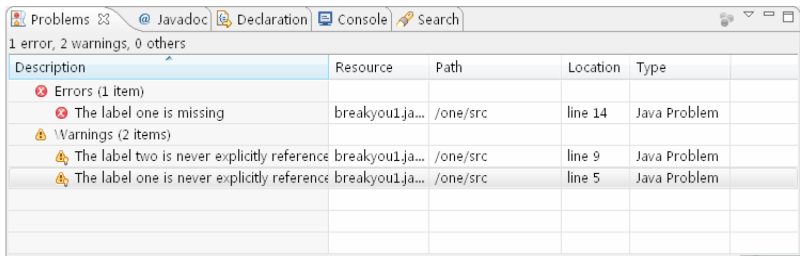
图 6-23　有标号的 break 语句

多学一招

程序运行后，先执行外层，再执行内层，输出 X=0；然后内层执行循环语句，输出 Y=0，接着依次输出 Y=1，Y=2，Y=3，Y=4。当输出 Y=7 时，将会执行 break 语句，退出 out 循环语句，退出循环。在使用有标号退出循环时，初学者很容易出现错误，下面将展示一个段错误的代码，其代码见“光盘：源代码/第 6 章/breakyou1.java”：

```
    public class breakyou1
    {
      public static void main(String args[])
      {
            one:for(int i=0;i<5;i++)
                {
                  System.out.println(i);
                }
            two:for(int j=0;j<5;j++)
                {
                  System.out.println(j);
                  if(j==3)
                  {
                        break one;
                  }
                }
      }
    }
```

运行代码，将会看到如图 6-24 所示的错误提示。



Description	Resource	Path	Location	Type
Errors (1 item)				
The label one is missing	breakyou1.ja...	/one/src	line 14	Java Problem
Warnings (2 items)				
The label two is never explicitly reference	breakyou1.ja...	/one/src	line 9	Java Problem
The label one is never explicitly reference	breakyou1.ja...	/one/src	line 5	Java Problem

图 6-24　提示的错误

提示：读者一定要注意，带标号的 break 语句只能放在这个标号所指的循环里面，否则将会出现编译错误。

6.2.2　return 语句的应用

return 语句通常是用来返回一个方法的值，并把控制权交给调用它的语句，其格式如下：

```
return[表达式];
```

参数介绍如下：

表达式是可选参数，表示要返回的值，它的数据类型必须同方法声明中的返回值类型一致，这可以通过强制类型转换实现。

在 Java 的程序设计里，return 语句是放在方法的最后，用于退出当前的程序，并返回一个值，当把单独的 return 语句放在一个方法的中间时，就会出现编译错误，如果用户要把 return 语句放在中间，可以使用条件语句 if，然后将 return 语句放在一个方法中间，用来实现在程序中未执行的全部语句退出，下面将通过一个实例进行讲解。

实例 24：认识 return 语句

下面将展示一个 return 语句，让读者认识 return 语句，其代码见“光盘：源代码/第 6 章/return1.java”：

```
public class return1
{
   public static int gcd(int a, int b)
{
         int min=a;
         int max=b;
         if (a > b)
 {
         min=b;
         max=a;
         }
         if (min==0)
         return max;
```

```
        else
        return gcd(min, max-min);
        }
    public static void main(String[] args)
  {
    System.out.println(return1.gcd(75, 15));
    }
  }
```

运行代码，得到如图 6-25 所示的结果。

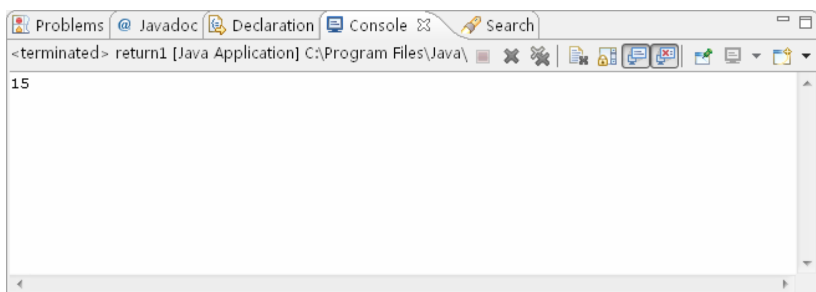


图 6-25　程序结果

多学一招

在上面的实例中，用户对 return 语句有了简单的认识，下面再展示一段代码，对 return 语句进行更深层次的学习，其代码见“光盘：源代码/第 6 章/return2.java”：

```
public class return2
{
    public static void main(String[] args)
    {
            Cat c1=new Cat(1,2,3);//实例化
            Cat c2=new Cat(1,2,3);
            System.out.println(c1.equals(c2));
            }
            }
            class Cat {
            int colour;
            int height,weight;
            public Cat(int colour,int height,int weight) {
            this.colour=colour;  //使用全局变量
            this.height=height;  //使用全局变量
            this.weight=weight;  //使用全局变量
            }
            public boolean equals(Object obj) {
            if (obj==null) return false;
            else
```

```
                    if(obj instanceof Cat) {
                    Cat c=(Cat)obj;
                    if((c.colour==colour)&&(c.height==height)&&(c.weight==
weight)) {
                    return true;
                    }
                    }
                    return false;
                    }
        }
```

运行代码，得到如图 6-26 所示的结果。

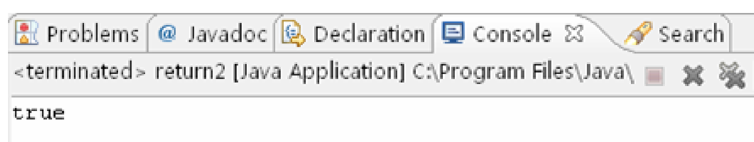


图 6-26　程序结果

6.2.3　continue 跳转语句

continue 跳转语句相对于前面的跳转语句应用得较少，它的作用就是强制一个循环提前返回。换句话说就是让循环继续执行，但不执行本次循环剩余的循环体中的语句。

实例 25：使用 continue 语句

continue 语句常出现在循环语句中，它的作用就是强制让循环语句提前返回，换句话说就是让循环继续跳过此次循环，进入下一次循，下面将展示一段代码，其代码见“关盘：源代码/第 6 章/conone.java”：

```
        public class conone
        {
           public static void main(String args[])
           {
                 for(int a=0;a<10;a++)
                 {
                       System.out.print(a);
                       if(a%2==0)
                       {
                             continue;
                       }
                       System.out.println("$");
                 }
           }
        }
```

运行代码，得到如图 6-27 所示的结果。

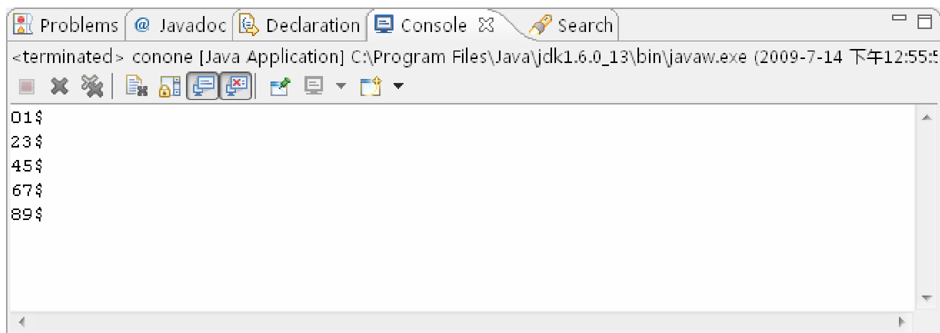


图 6-27　运行结果

多学一招

对于上面的程序其实很好理解，先进入循环，输出 0，然后执行控制语句，计算结果为 true ，执行 continue 语句，则再也不执行循环语句中剩余语句。回到循环语句，输出 1，然后进入选择控制语句，计算结果为 false ，则不再执行 continue 语句。继续执行并输出美元符号（$），依次类推。这个代码是无标号的，continue 也可带标号，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 6 章/contwo.java”如下：

```
public class contwo
 {
   public static void main(String args[])
   {

         out:for(int a=1;a<=9;a++)
             {
               for(int b=1;b<=9;b++)
               {

                     if(b>a)
                     {
                           System.out.println();
                           continue out;
                     }
                     System.out.print("  "+a+"*"+b+"="+a*b);
               }
               System.out.println();
             }
   }
}
```

运行代码，得到如图 6-28 所示的结果。

提示：读者可将这里的九九乘法表和本章开篇的作比较，并思考两者的异同。

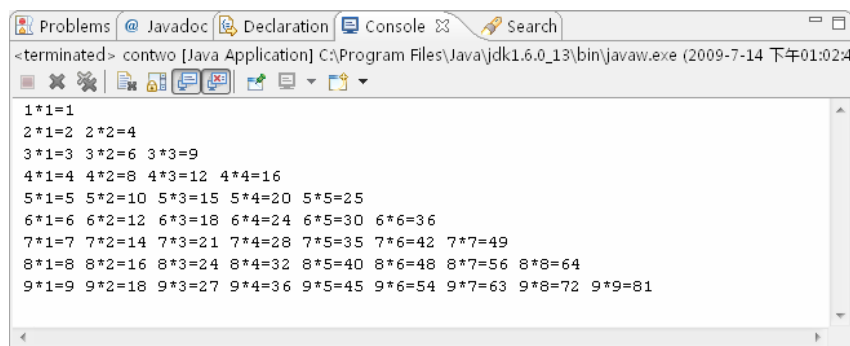


图 6-28 continue 语句

6.2.4 轻松使用跳转语句

至此，读者已经学习了三个跳转语句，但究竟什么时候该用哪种跳转语句最好，对于初学者来说是一道难题。

代码 31：下面展示一段代码，其代码见“光盘：源代码/第 6 章/tiao.java”：

```
public class tiao {
   public static void main(String[] args)
   {
         int i=0;
         outer:
               while(true)
               {i++;
               inner:
                     for(int j=0;j<15;j++)
                     {
                           i+=j;
                           if(j==3)
                                 continue inner;
                           break outer;
                     }
               continue outer;
                     }
         System.out.println(i);
   }
}
```

这段代码执行的结果很简单，只显示一个“1”。其中用到了 break 和 continue 语句，它们都是用来停止循环语句的。但两者有一定的区别，break 是用来停止整个循环的，并开始处理 break 程序块的后面一行代码，而 continue 语句只是用于停止当前循环，并开始执行同一循环的下一次循环的。

下面再展示一段代码，其代码见“光盘：源代码/第 6 章/tiao1.java”：

```
public class tiao1 {
   final static int Aa=10;
   public static void main(String[] args){
         for(int Bb=0;Bb<Aa;Bb++){
               System.out.print(Bb);
               if(Bb>5){
                     break;
               }
               System.out.print(Bb);
   }
}
}
```

运行代码，得到如图 6-29 所示的结果。

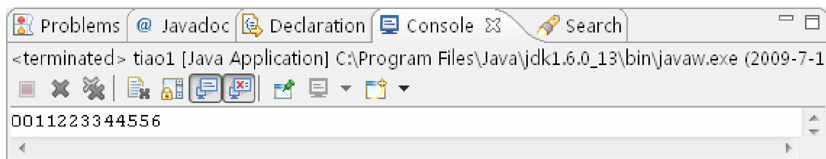


图 6-29　执行结果

用户可对上面的代码进行修改，将“break;”修改成“continue;”，其结果将会产生变化，如图 6-30 所示。

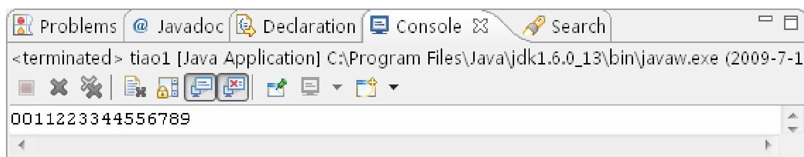


图 6-30　输出结果

6.3　疑难问题解析

本章详细介绍了循环语句、跳转语句等基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：for 语句嵌套循环功能很强大，while 语句和 do⋯while 语句可以实现嵌套功能吗？

解答：可以，但是 for 语句嵌套使用最为简单，功能也最为强大，若使用 do…while 语句和 while 语句嵌套，结构将变得很复杂。

读者疑问：for 语句、do⋯while 语句和 while 语句，这三个循环语句可以混合使用吗？

解答：从语法上讲是可以的，在 for 语句中可以使用 while 语句，在 do⋯while 语句中也

可以使用 for 语句，但在程序中，很少见到这类程序，因为几种循环的混用，容易造成结构混乱。



职场点拨——面试的准备

面试是我们步入职场的敲门砖，只有面试成功后，才能真正进入到一家企业去上班。都说不打无把握之仗，在面对即将开始的面试前，你知道需要准备什么吗？综合考虑之下，你必须做好如下 4 种准备。

（1）心理准备

谁在面试时都会紧张，笔者本人也不例外。紧张的原因是多方面的，但最关键的原因就在于自己不自信，在面试时顾虑重重。你可能会担心不知道面试官会问你什么问题，也可能不知道自己会不会回答得不够得体，还可能你不知道前后的应聘者会不会表现得比你更优秀……确实，对于刚接到面试通知的你来说，一切都是未知数。但是请记住一点，把你所能够掌控的准备到最充足，那么和其他的面试者相比，你就有了更多的胜算，你也就会更自信。机会是给有准备的人的，这句话永远也不会错。

我建议读者在面试时一定要保持轻松的心态，发挥“战略上要藐视、战术上要重视”的优良传统。实践证明面试时的姿态和表情非常重要，面试时我们要一直面带微笑，表现出真诚和自信，相信你一定会收获满意的结果。

（2）资料准备

面试之前一定要准备好需要的资料，例如毕业证书和学位证书、英语等级证书、个人简历、户口簿第一页及毕业证复印件、原公司名称、地址、证明人等。如果单位有要求，还要带着离职证明。

（3）提前了解公司的资料

都说不打无把握之仗，我们在去对方公司面试之前，可以提前登录对方的官方网站，或者利用网络和报纸等媒体，对这家公司做一个充分了解。预先了解这个公司一般是做哪部分业务的，究竟是做 OA 系统、电子商务系统还是 Web 系统。

（4）问题准备

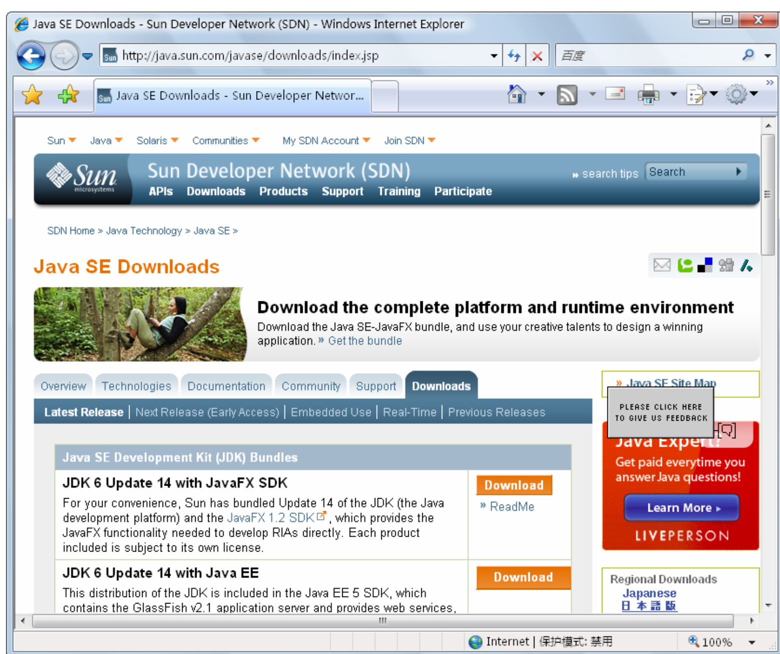
面试时肯定会面对考官提出的问题，但是你也可以向对方提出问题。因为应聘是一个双向选择过程，不仅仅是用人单位在单方面选择你，你同样也在选择合格的公司。你很有必要询问公司的发展趋势、市场开拓情况、为什么要招聘这个职位、公司的用人标准、管理风格等可能会对我们发展有影响的实际问题。

温故而知新——第一篇实战范例

第一篇的内容十分简单，主要是讲解 Java 的运行环境、Java 的开发环境和 Java 的基础语法。在这里，将对本篇内容进行回顾，以让用户熟练掌握本篇内容。

范例 1　获得 JDK

JDK 是支持开发和运行 Java 的重要软件，没有它，用户将无法开发 Java。JDK 有许多版本，这里下载 Sun 公司最新版本的 JDK（除了 Sun 公司的 JDK 外，IBM 也有自己的 JDK，但普遍使用的是 Sun 公司的 JDK），用户可以去 sun 公司官方网站下载 JDK，如范例图 1-1 所示。



范例图 1-1　下载 JDK

范例 2　配置运行环境

安装好 JDK 后，用户需要对自己的计算机的环境进行配置，配置的方法十分简单，用户只需要新建几个环境变量即可。配置完成后，单击开始菜单，在“运行”文本框输入 cmd 进行 dos 命令界面，然后输入 javac，得到如范例图 1-2 所示的结果。

```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.0.6001]
版权所有 (C) 2006 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>javac
用法: javac <选项> <源文件>
其中, 可能的选项包括:
  -g                         生成所有调试信息
  -g:none                    不生成任何调试信息
  -g:{lines,vars,source}     只生成某些调试信息
  -nowarn                    不生成任何警告
  -verbose                   输出有关编译器正在执行的操作的消息
  -deprecation               输出使用已过时的 API 的源位置
  -classpath <路径>            指定查找用户类文件和注释处理程序的位置
  -cp <路径>                   指定查找用户类文件和注释处理程序的位置
  -sourcepath <路径>           指定查找输入源文件的位置
  -bootclasspath <路径>        覆盖引导类文件的位置
  -extdirs <目录>              覆盖安装的扩展目录的位置
  -endorseddirs <目录>         覆盖签名的标准路径的位置
  -proc:{none,only}          控制是否执行注释处理和/或编译。
  -processor <class1>[,<class2>,<class3>...]要运行的注释处理程序的名称; 绕过默认
的搜索进程
  -processorpath <路径>        指定查找注释处理程序的位置
  -d <目录>                    指定存放生成的类文件的位置
  -s <目录>                    指定存放生成的源文件的位置
  -implicit:{none,class}     指定是否为隐式引用文件生成类文件
  -encoding <编码>             指定源文件使用的字符编码
  -source <版本>               提供与指定版本的源兼容性
  -target <版本>               生成特定 VM 版本的类文件
  -version                   版本信息
  -help                      输出标准选项的提要
  -Akey[=value]              传递给注释处理程序的选项
  -X                         输出非标准选项的提要
  -J<标志>                     直接将 <标志> 传递给运行时系统


C:\Users\Administrator>
```

范例图 1-2　成功配置 java 运行环境

提示：执行 javac 命令后，用户可以清楚看到一些命令及其后面相应的提示，用户不妨记住这些命令，并多加练习以熟悉 Java 语言的调试。

范例 3　安装 Java 的开发工具

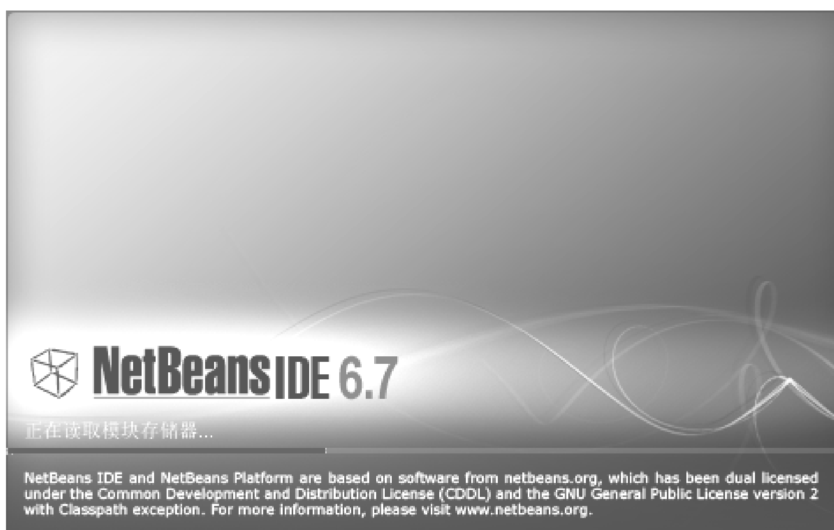
开发 Java 程序的工具主要有两种，一种是 NetBeans，一种是 Eclipse，这里将对这两个软件进行简单回顾，如范例图 1-3 为 Netbeans 的启动界面。

提示：Eclipse 本身功能并不强大，但它有很多优秀的插件，如 MyEclipse 插件就是它最好的插件之一，用它可以开发优秀的 JavaEE 系统。

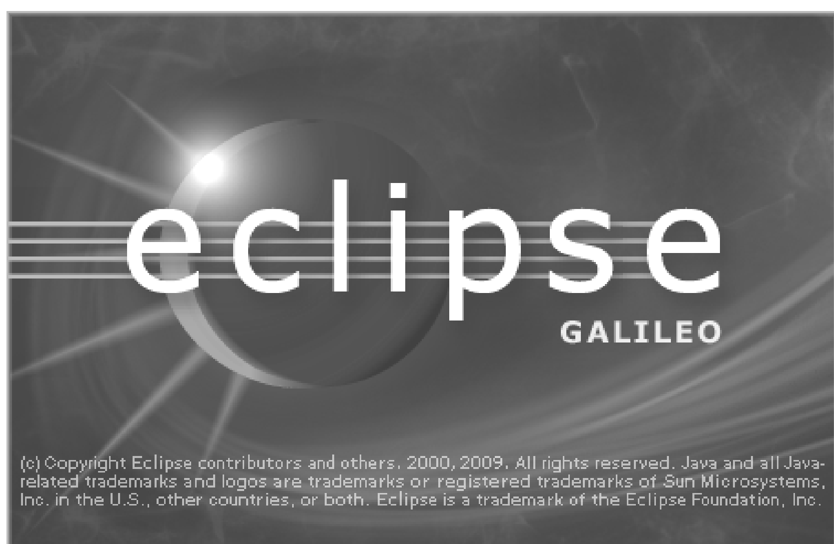
Eclipse 是不需要安装的，下载后，将其解压就可以使用，启动界面如范例图 1-4 所示。

范例 4　量、数据类型

在 Java 中，量的重要性不言而喻，量分为常量和变量，常量和变量都必须定义自己的数据类型。下面通过一段代码对量和数据类型进行回顾，其代码见“光盘：源代码/温故而知新 1/ Liang.java”：



范例图 1-3　NetBeans 启动界面



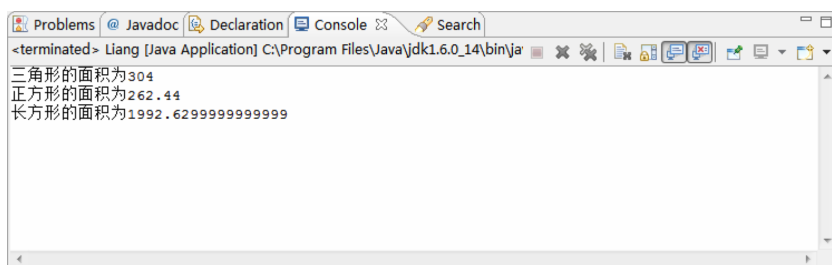
范例图 1-4　Eclipse 启动界面

```
public class Liang
{
   public static void main(String args[])
   {
          //三角形面积
         int x=38,y=16;
         int z=x*y/2;
         System.out.println("三角形的面积为"+z);

//正方形面积
```

```
            double a1=16.2;
            double s1=a1*a1;
            System.out.println("正方形的面积为"+s1);
          //长方形面积
            double a2=38.1,b2=52.3;
            double s2=a2*b2;
            System.out.println("长方形的面积为"+s2);
            }
    }
```

运行代码，得到如范例图 1-5 所示的结果。



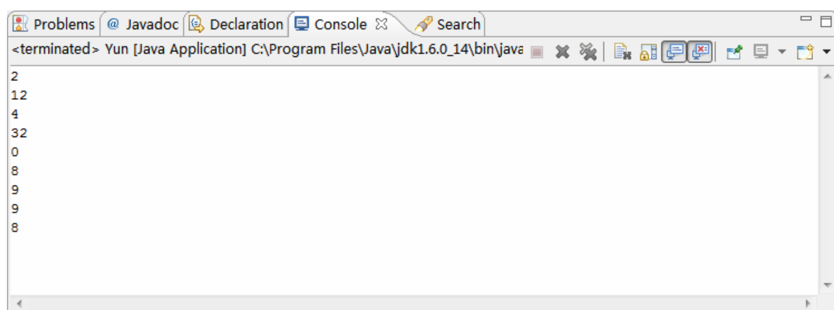
范例图 1-5　执行结果

范例 5　运算符

对任何一种程序来说，运算符的作用都是不可小视的，它可以用来计算任何数据类型，以得到想要的效果。下面通过一段代码进行回顾，其代码见“光盘：源代码/温故而知新 1/Yun.java”：

```
    public class Yun
    {
       public static void main(String args[])
    {
            int a=8;
            int b=4;
            System.out.println(a/b);
            System.out.println(a+b);
            System.out.println(a-b);
            System.out.println(a*b);
            System.out.println(a%b);
            System.out.println(a++);
            System.out.println(a--);
            System.out.println(++a);
            System.out.println(--a);
       }
    }
```

运行代码，得到如范例图 1-6 所示的结果。



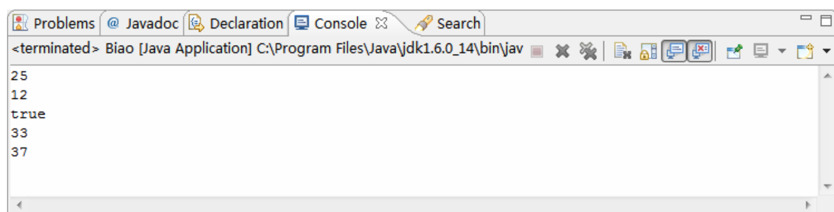
范例图 1-6　运算符

范例 6　表达式

在任何一个非空 Java 程序里都有表达式，用户一定理解表达式的概念。下面通过一段代码进行回顾其代码见“光盘：源代码/温故而知新 1/Biao.java”：

```
public class Biao
{
   public static void main(String[] args)
   {
               int aa=31;
               int bb=2;
             int hh=8;
             int kk=4;
             int xx=aa-bb+kk-hh;
             int yy=bb*kk/bb+hh;
             int zz=aa+bb;
             int b2=kk+zz;
           System.out.println(xx);
           System.out.println(yy);
           System.out.println(xx!=yy);
           System.out.println(zz);
           System.out.println(b2);
         }
}
```

运行代码，得到如范例图 1-7 所示的结果。



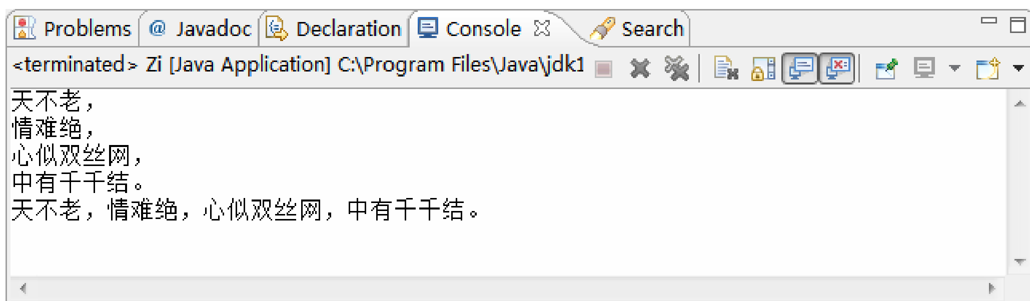
范例图 1-7　表达式

范例 7　字符串

在 Java 程序中，有许多方法可以操作字符串，如在串中查找某个子串、求取一个子串的长度、在串的某个位置上插入一个子串以及在字符串中删除一个子串等，下面通过一段代码来讲解字符串的基本操作，其代码见“光盘：源代码/温故而知新 1/Zi.Java”：

```
public class Zi
{
   public static void main(String[] args)
   {
               String A=new String("天不老，");
               String  K="情难绝，";
               String  Q="心似双丝网，";
               String  J="中有千千结。";
             String E=A+K+Q+J;
             System.out.println(A);
             System.out.println(K);
             System.out.println(Q);
             System.out.println(J);
             System.out.println(E);
         }
}
```

运行代码，得到如范例图 1-8 所示的结果。



范例图 1-8　字符串

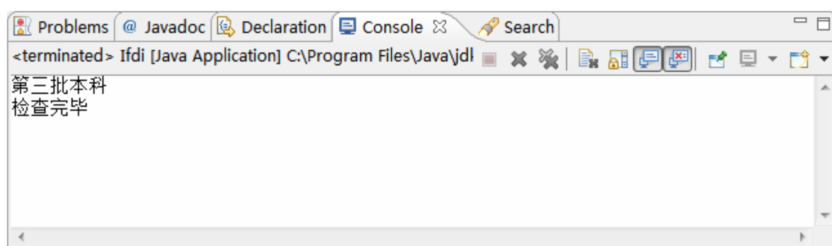
范例 8　if 语句

if 语句是假设语句，换句话说，if 语句是最基础的条件语句。在 Java 程序中，它有着举足重轻的地位。下面通过一段代码进行回顾，其代码见“光盘：源代码/温故而知新 1/Ifdi.java”：

```
public class Ifdi
{
               public static void main(String args[])
         {
```

```
                int 成绩=532;
                if(成绩>610)
                     System.out.println("考上重点本科线");
                else if(成绩>570)
                  System.out.println("一般本科线");
                else if(成绩>520)
                       System.out.println("第三批本科");
                else if(成绩>450)
                  System.out.println("专科线");
                else if(成绩>390)
                       System.out.println("高职专科线");
                else
                  System.out.println("没有上线");
                System.out.println("检查完毕");
           }
    }
```

运行代码，得到如范例图 1-9 所示的结果。



范例图 1-9　if 语句

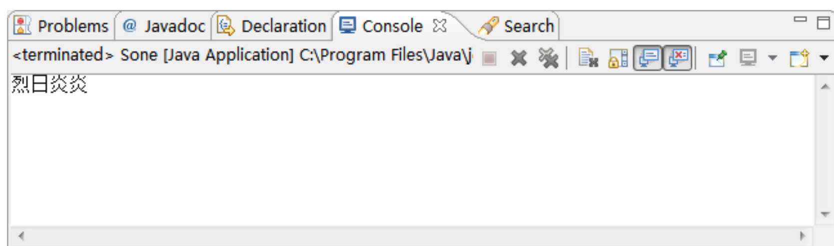
范例 9　switch 语句

switch 语句方法就是为了判断多条件而诞生的，它的使用方法和 if 嵌套语句十分相似。下面通过一段代码进行回顾，其代码见“光盘：源代码/温故而知新 1/Sone.java”：

```
    public class Sone
     {
       public static void main(String args[]){
           int month=7;
           switch(month){
                case 12:
                case 1:
                case 2:
                  System.out.println("寒冬日月");
                  break;
                case 3:
                case 4:
                case 5:
```

```
                    System.out.println("春暖花开");
                    break;
                case 6:
                case 7:
                case 8:
                    System.out.println("烈日炎炎");
                    break;
                case 9:
                case 10:
                case 11:
                    System.out.println("秋叶纷飞");
                    break;
                default:
                    System.out.println("输入错误");
            }
        }
    }
```

运行代码，得到如范例图 1-10 所示的结果。



范例图 1-10　switch 语句

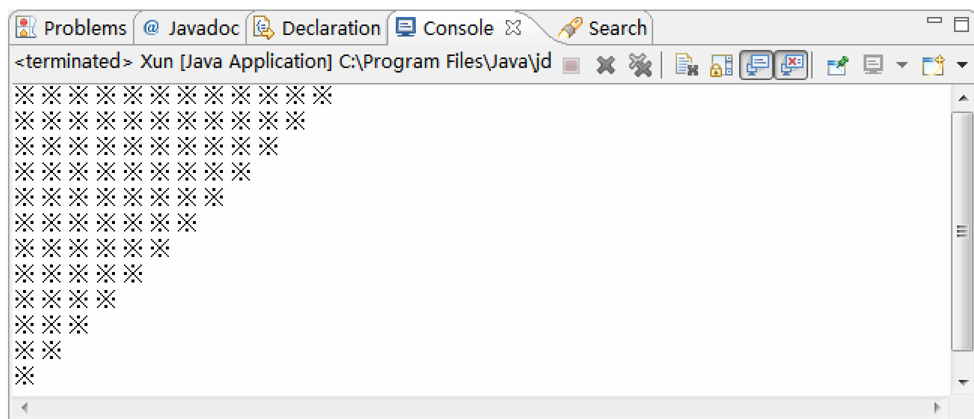
范例 10　for 循环语句

for 语句是最为常见的一种循环语句，for 循环是一个功能强大且形式灵活的结构。下面通过一段代码进行回顾，其代码见“光盘：源代码/温故而知新 1/Xun.java”：

```
public class Xun
 {
   public static void main(String[] args)
    {
     for(int a=12; a>=1; a--)
     {
      for(int b=a; b>=1; b--)
      {
      System.out.print("※" + " ");
       }
       System.out.println();
     }
```

```
        }
    }
```

运行代码，得到如范例图 1-11 所示的结果。



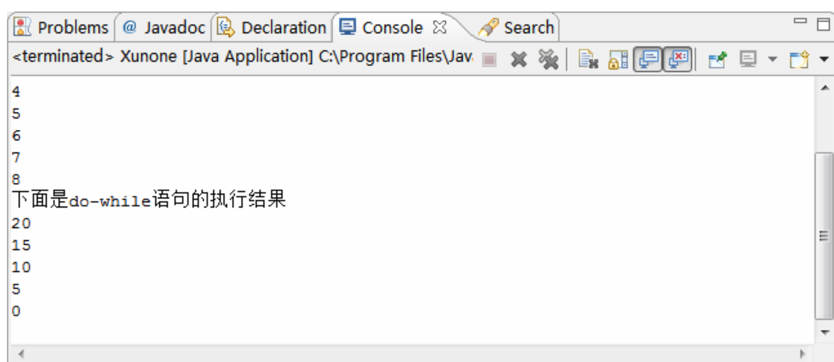
范例图 1-11　for 循环语句

范例 11　while 和 do…while 循环语句

while、do…while 语句和 for 循环语句的功能大致相同，但在结构上有很大不同。下面通过一段代码进行回顾，其代码见“光盘：源代码/温故而知新 1/Xunone.java”：

```
    public class Xunone
    {
       public static void main(String[] args)
       {
             int sum=0;
             int a=1;      //由于是计算自然数，所以 1 的初始值设置为 1
             System.out.println("累加和不大于 30 的所有自然数如下：");
             while(sum<30){
                   sum=sum+a;
                   System.out.println(a);
                   a++;      //该语句一定不用少
             }
             System.out.println("下面是 do-while 语句的执行结果");
             int ting=25;
             do
             {
             ting=ting-5;
             System.out.println(ting);
             }
             while (ting>=5);
       }
    }
```

运行代码，得到如范例图 1-12 所示的结果。



范例图 1-12　while 语句和 do…while 语句

范例 12　数组

数组在程序设计中是最为常见的一种数据结构，它是将相同类型的数据用一个标识符来封装到一起的对象序列或基本类型序列，它的存储效率最高。下面通过一段代码进行预习，其代码见“光盘：源代码/温故而知新 1/shuzu.java”：

```
public class shuzu
{
  public static void main(String[] args)
  {
       int[][] X={
                  {1,2,3,4},
                  {5,6,7,8},
                  {9,10,17,18},
                  {15,19,25,26},
                  {37,18,39,40},
       };

       int[][] Y={
                  {1,2,3,4},
                  {5,6,7,8},
                  {9,0,7,8},
                  {3,4,5,6},
                  {1,8,9,20},
       };
       Y=X;
       for(int i=0;i<X.length;i++)
            for(int j=0;j<X[i].length;j++)
{
                  System.out.println("X["+i+"]["+j+"] ="+X[i][j]);
                  System.out.println("Y["+i+"]["+j+"] ="+Y[i][j]);
```

```
            }
        }
    }
```

运行代码，得到如范例图 1-13 所示的结果。

```
Problems  @ Javadoc  Declaration  Console  Search
<terminated> shuzu [Java Application] C:\Program Files\Java\
X[1][3] =8
Y[1][3] =8
X[2][0] =9
Y[2][0] =9
X[2][1] =10
Y[2][1] =10
X[2][2] =17
Y[2][2] =17
X[2][3] =18
Y[2][3] =18
X[3][0] =15
Y[3][0] =15
X[3][1] =19
```

范例图 1-13　数组

第二篇　核心技术篇

第 7 章　特殊数据——数组

在程序设计里，有一种数据十分特殊。它就是数组。数组在程序设计里是最为常见的一种数据结构。它是将相同类型的数据，用一个标识符来封装到一起的对象序列或基本类型序列，它的存储效率最高。本章将详细讲解数组和数组的基本操作。本章主要内容如下：

- 一维数组。
- 二维数组。
- 多维数组。
- 数组操作。
- 职场点拨——客户沟通之道。

2010 年 X 月 X 日，天气阴转晴

有许多师姐说，学习程序，数组是比较复杂、难以理解的，什么是数组呢？它在程序中有什么作用？哪位师兄能指点一二？



一问一答

小菜：“Wisdom，数组和字符串究竟有什么作用？为什么要用它来包装变量和常量？不是很明白。”

Wisdom：“呵呵，当时我学习时也有这个疑问。现在我明白了，在编程过程中同一类型的变量和常量经过包装后，会以一种新的面貌展示，功能更强大，编程更简单。”

小菜：“数组重要吗？”

Wisdom：“当然重要了！数组和字符串是 Java 中重要的组成部分，数组属于构造数据类型。一个数组可以拥有多个数组元素，这些数组元素可以是基本数据类型或是构造类型。因此按数组元素的类型不同，数组又可分为数值数组、字符数组、指针数组、结构数组等各种类型。”

7.1　简单的一维数组

数组中以一维数组最为常见，使用它也最为普遍，那如何定义一个一维数组呢？如何对它进行初始化呢？本节将详细讲解。

7.1.1　声明一维数组

数组就是元素的集合体，每一个元素都拥有一个索引值，只需要指定索引值就可以取出对应的数据。下面讲解一维数组的声明格式，其格式如下：

```
int[] array;
```

或者

```
int array[];
```

这两种格式只是形式不同，但含义是一样的。下面讲解参数，其参数介绍如下：

- int：数组元素类型
- array：数组名称
- [ ]：一维数组的内容都是由这个符号括起来的。

除了上面的声明整型数组外，还可以声明多种数据类型的数据，例如：

```
boolean[] array;        //声明布尔数组
float[] array;          //声明浮点数组
double[] array;         //声明双精度数组
```

7.1.2　创建一维数组

所谓创建数组实质上就是为数组申请相应的存储空间，数组的创建需用大括号（{}）括起来，然后将一组相同类型的数据放在存储空间里，存储空间如何分配由 Java 编译器负责管理。创建数组的方法十分简单，其格式如下：

```
int[] a={1,2,3,5,8,9,15};
```

这个数组是一个简单的整形数组，数组名为 a，但是为了访问数组中的特点元素，应指定数组的元素的位置序数，也就是索引和下标。一维数组具体结构如图 7-1 所示。

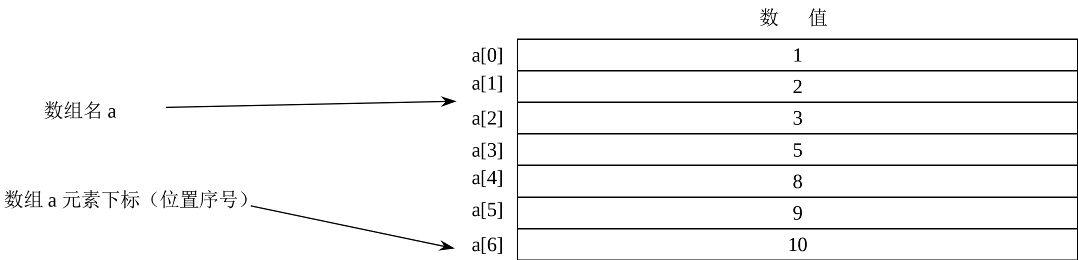


图 7-1　一维数组内部结构

结构图中数组的名称是 a，“［]”中的值为序号（下标），这样就可以很清楚地表示每一个数组元素。a 数组的第一个值就用 a[0]表示，第 2 个值就用 a[1]，以此类推。上面一节中讲到的公司员工工资的问题就可以通过这种方法，用 a[0]、a[1] 、a[2] 、a[3]……a[650]来表示。

实例 26：使用一维数组

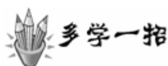
下面通过一段代码讲解一维数组，其代码见“光盘：源代码/第 7 章/shuzuone1.java”：

```
public class shuzuone1
 {
   public static void main(String[] args)
   {
//定义数组
         int[] X={12,13,24,77,68,39,60};
         int[] Y;
         Y=X;
         for(int i=0;i<X.length;i++)
         {
               Y[i]++;
               System.out.println("X["+i+"]="+X[i]);
               System.out.println("Y["+i+"]="+Y[i]);
         }
   }
}
```

运行代码，得到如图 7-2 所示的结果。

```
Problems  @ Javadoc  Declaration  Console  Search
<terminated> shuzuone1 [Java Application] C:\Program Files\Java\jdk1.6.0_13\bin\javaw.exe (2009-7-14 下午02:10:47)
X[0]=13
Y[0]=13
X[1]=14
Y[1]=14
X[2]=25
Y[2]=25
X[3]=78
Y[3]=78
X[4]=69
Y[4]=69
X[5]=40
Y[5]=40
X[6]=61
Y[6]=61
```

图 7-2　一维数组



数组计数都是从零开始的，最大数组下标为“length-1”。在上面的程序中，数组 Y 没有任何元素，它只是被实例化了的一个对象，用于告诉编译器为它分配一定的存储空间。然后数组 X 赋值给 Y，这个编译操作实际上就是将 X 数组的内存地址赋给数组 Y。在实例中，Y 数组并没有赋值，下面通过另一段代码将 Y 数组也赋值，其代码见“光盘：源代码/

第 7 章/shuzuone2.java”：

```
public class shuzuone2
{
   public static void main(String[] args)
   {
                  int[] X={12,13,24,77,68,39,60};
         int[] Y={22,23,25,79,78,1,56,23,67,100};
         Y=X;
         for(int i=0;i<X.length;i++)
         {
               Y[i]++;
               System.out.println("X["+i+"]="+X[i]);
               System.out.println("Y["+i+"]="+Y[i]);
         }
   }
}
```

运行代码，得到如图 7-3 所示的结果。

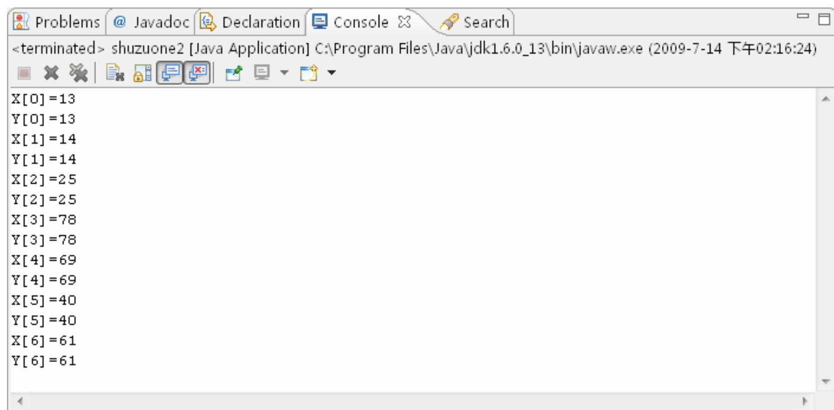


图 7-3　一维数组

提示：上面的多学一招中的程序实现了和实例一样的功能，其原因是它们都有一行代码：Y=X。这个代码的功能就是将数组 X 的值赋值给数组 Y。如果将这行代码去除，运行上面的程序将会得到如图 7-4 所示的结果。

7.1.3　轻松初始化一维数组

在 Java 程序里，一定要将数组看作一个对象，它的数据类型和前面的基本数据类型相同，用户有时需要对它进行初始化。在初始化的时候，用户可以规定数组的大小，当然也可以初始化数组中的每一个元素。下面讲解初始化一维数组的方法，格式如下：

```
int[] a=new int[8];
int[] a=new int{1,2,3,4,5,6,7,8};
int[] a={1,2,3,4};
```

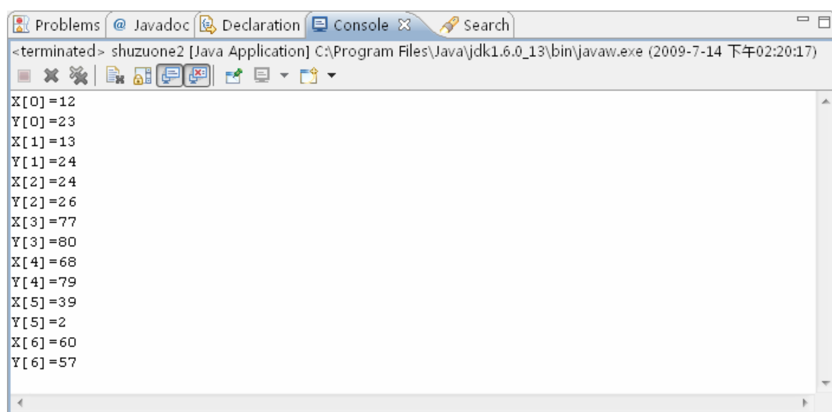


图 7-4　两个不同的数组

参数介绍如下：

- int：数组类型。
- a：数组名称。
- new：对象初始化语句。

提示：在初始化数组的时候，使用 new 创建数组后，一定要知道它只是一个“引用”，直到将值赋给“引用”，初始化进行才算真正结束。在上面三种初始化数组的方法中，用户可以根据自己的习惯选择其中的一种。

实例 27：初始化一维数组

理解了一维数组后，下面将通过一段代码讲解一维数组，其代码见“光盘：源代码/第 7 章/shuzuone3.java”：

```
import java.util.Random;//插入 Random 包
public class shuzuone3{
   public static void main(String[] args) {
                    Random rand=new Random();  //实例化 Random 类对象
int[]x=new int[rand.nextInt(12)];  //随机产生 0-20 之间的数作为 int 数组的长度
double[]y=new double[rand.nextInt(12)];
//随机产生 0-20 之间的数作为 int 数组的长度
        System.out.println("x 的长度为"+x.length);
        System.out.println("y 的长度为"+y.length);
        for(int i=0;i<x.length;i++){
              x[i]=rand.nextInt(12);      //随机产生 0-20 之间数的赋给数组 a
              System.out.println("x["+i+"]="+x[i]); //打印数组 a
        }
        for(int i=0;i<y.length;i++){
              y[i]=rand.nextDouble();     //随机产 double 数的赋给数组 b
              System.out.println("y["+i+"]="+y[i]);//打印数组 b
        }
   }
}
```

运行代码，得到如图 7-5 所示的结果。

```
Problems  @ Javadoc  Declaration  Console ☒  Search
<terminated> shuzuone3 [Java Application] C:\Program Files\Java\jdk1.6.0_13\bin\javaw.exe (2009-7-14 下午02:32:35)
x的长度为11
y的长度为9
x[0]=4
x[1]=4
x[2]=0
x[3]=7
x[4]=3
x[5]=4
x[6]=10
x[7]=10
x[8]=0
x[9]=5
x[10]=10
y[0]=0.3782913430575201
y[1]=0.6783741843451766
y[2]=0.652003031928813 5
y[3]=0.14863322020121683
y[4]=0.20706950457376383
y[5]=0.6876855384294285
y[6]=0.339466498271512 1
y[7]=0.03156165454350146
y[8]=0.041909279481005424
```

图 7-5　执行结果

多学一招

在上面的实例中初始化了一维数组，然后随机将其打印在下面的程序中，将初始化两个不同类型的数组，然后对其元素进行运算，其代码见“光盘：源代码/第 7 章/shuzuone3.java”：

```
import java.lang.Math;
class Array{
   public Array(int i){
         System.out.println("int 型数组"+i);
   }
   public Array(double i){
         System.out.println("double 型数组"+i);
   }
   public Array(int i,int j){
         System.out.println("相乘数组"+i*j);
   }
}
public class shuzuone4{
   public static void main(String[] args) {
         Array[] ArrayB=new Array[5];
         int [] ArrayC=new int[5];
         double[] ArrayD=new double[5];
         for(int i=0;i<ArrayC.length;i++){
               ArrayC[i]=i;
```

```
                ArrayB[i]=new Array(ArrayC[i]);}
            for(int i=0;i<ArrayD.length;i++){
                ArrayD[i]=Math.atan(i);
                ArrayB[i]=new Array(ArrayD[i]);
            }
            for(int i=0,j=0;i<ArrayC.length;i++,j++){
                ArrayB[i]=new Array(ArrayC[i],ArrayC[j]);
            }
        }}
```

运行代码，得到如图 7-6 所示的结果。

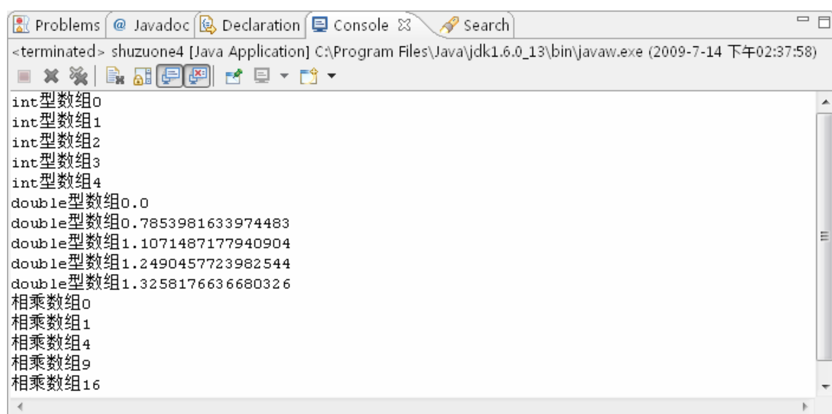


图 7-6　初始化数组

7.2　二维数组

在 Java 程序设计里，二维数组应用最为广泛。什么是二维数组？所谓二维数组就是有两个底标的数组，初学者可以将二维数组理解成一个围棋棋盘，要描述一个元素的位置，必须通过纵横两个底标来描述。在本节的学习中，将详细讲解二维数组。

7.2.1　二维数组的声明

在前面学习了一维数组的声明，实际上二维数组和一维数组十分相似。很多程序程序设计者都习惯将二维数组看成一个特殊的一维数组，其每一个元素又是一个数组。声明二维数组的方法十分简单，其格式如下：

```
float A[][];
char B[][];
int C[][];
```

参数介绍如下：

- ❑ float：数组类型。
- ❑ A：数组名称。

- char：数组类型。
- B：数组名称。
- 数组 A 的元素可以存放在 float 型数据。

7.2.2　二维数组的创建

创建二维数组实际上就是在电脑中申请一个存储空间，如下面的二维数组：

```
Int A[][]=
{1,3,5,7},
{2,4,6,8};
```

上面是一个二维数组，A 为数组名，实际上该二维数组相当于一个两行四列的矩阵，当需要取多维中的值时，可以使用下标来显示。其格式如下：

```
Array[i-1][j-1]
```

参数介绍如下：

- i：数组的行数。
- j：数组的列数。

下面以一个二维数组为例，读者看一下 3 行 4 列的数组内部结构，如表 7-1 所示：

表 7-1　二维数组内部结构表

	列 1	列 2	列 3	列 4
行 0	A[0] [0]	A[0] [1]	A[0] [2]	A[0] [3]
行 1	A[1] [0]	A[1] [1]	A[1] [2]	A[1] [3]
行 2	A[2] [0]	A[2] [1]	A[2] [2]	A[2] [3]

实例 28：创建一个二维数组并将其打印

下面创建一个二维数组，然后将其打印，其代码见“光盘：源代码/第 7 章/shuzutwo1.java”：

```
public class shuzutwo1
{
   public static void main(String[] args)
   {
         int [][] Aa={
                     {11,12,23,24},
                     {15,26,27,18},
                     {19,10,17,18},
                     {13,14,15,16},
                     {17,18,19,20},
         };
         for(int i=0;i<Aa.length;i++)
               for(int j=0;j<Aa[i].length;j++)
{
```

```
                    System.out.println("Aa["+i+"]["+j+"] ="+Aa[i][j]);
                }
        }
    }
```

运行代码，得到如图 7-7 所示的结果。

```
Problems  @ Javadoc  Declaration  Console  Search
<terminated> shuzutwo1 [Java Application] C:\Program Files\Java\jdk1.
Aa[0][0] =11
Aa[0][1] =12
Aa[0][2] =23
Aa[0][3] =24
Aa[1][0] =15
Aa[1][1] =26
Aa[1][2] =27
Aa[1][3] =18
Aa[2][0] =19
Aa[2][1] =10
Aa[2][2] =17
Aa[2][3] =18
Aa[3][0] =13
Aa[3][1] =14
Aa[3][2] =15
Aa[3][3] =16
Aa[4][0] =17
Aa[4][1] =18
Aa[4][2] =19
Aa[4][3] =20
```

图 7-7　执行结果

提示：在程序中用到了 for 循环语句，在打印二维数组时，第一个 for 循环语句表示以行进行循环，第二个循环语句以每行的列数进行循环，这样就达到了取得二维数组中的每个值的功能。

多学一招

在上面的代码中只创建了一个二维数组，在下面这段代码中，将创建两个二维数组，然后将其中一个数组的值赋值到另外一个数组，其代码见“光盘：源代码/第 7 章/shuzutwo2.java”：

```
public class shuzutwo2 {
    public static void main(String[] args)
    {
        int[][] X={
                {11,12,23,24},
                {15,26,27,18},
                {19,10,17,18},
                {13,14,15,16},
                {17,18,19,20},
        };

        int[][] Y={
                {1,2,3,4},
                {5,6,7,8},
```

```
                        {9,0,7,8},
                        {3,4,5,6},
                        {1,8,9,20},
               };
               Y=X;
               for(int i=0;i<X.length;i++)
                    for(int j=0;j<X[i].length;j++)
        {
                        System.out.println("X["+i+"]["+j+"] ="+X[i][j]);
                        System.out.println("Y["+i+"]["+j+"] ="+Y[i][j]);
                    }
          }
        }
```

运行代码，得到如图 7-8 所示的结果。

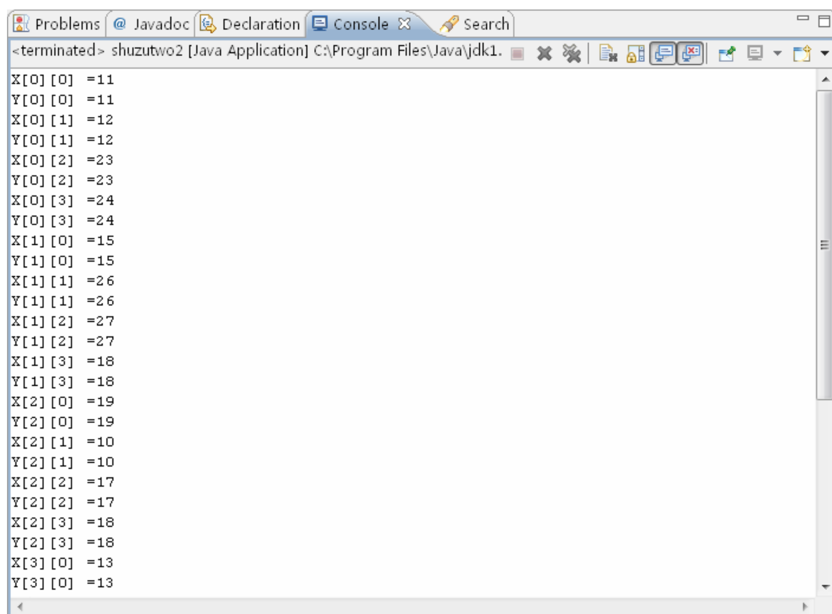


图 7-8　赋值后的二维数组

7.2.3　二维数组的初始化

二维数组的初始化很简单，学习过一维数组的读者朋友，理解起来就更为简单，因为它的初始化和一维数组的初始化一样，基本上也是使用下列语法来指定元素初始值，其语法格式如下：

```
array=new int[]…[]{第一个元素的值，第二个元素的值，第三个元素的值，…};
```

或用于对象数组的语法：

```
array=new int[]…[]{new 构造方法（参数列），{new 构造方法（参数列），…};
```

其参数介绍如下：

- array：数组名称。
- new：实例化对象语句。
- int：数组元素类型。

二维数组是多维数组，为了结构清晰，必须使用多个“{}”括起来。下面以二维数组为例，若希望第一维有 3 个索引，第二维有两个索引，就要依下列语法来指定元素的初始值，例如：

```
integer[][]array=new Integer[][]{
{new Integer(1), new Integer(2)},
{new Integer(3), new Integer(4)},
{new Integer(5), new Integer(6)},
}
```

参数介绍如下：

- array：数组名称。
- int：数组元素类型。
- new：实例化对象语句。
- Integer：数组元素类型。

实例 29：初始化一个二维数组

下面初始化一个二维数组，并输出其中的最大数和最小数，其代码见“光盘：源代码/第 7 章/shuzutwo3.java”：

```
public class shuzutwo3
 {
   int grades[][]={
            {37,78,96,43},
            {26,17,99,11},
            {40,90,86,81}
   };
   public static void main(String args[]){
        shuzutwo3 m=new shuzutwo3();
        System.out.println("最小的数："+m.minimum());
        System.out.println("最大的数："+m.maximum());
            }
   public int minimum()
{
        int lowGrade=grades[0][0];
        for(int row=0;row<grades.length;row++)
{
            for(int column=0;column<grades  [row].length;column++)
{
                if(grades[row][column]<lowGrade)
                    lowGrade=grades[row][column];
            }
```

```
            }
            return lowGrade;
        }
        public int maximum()
    {
            int highGrade=grades[0][0];
            for(int row=0;row<grades.length;row++)
    {
                for(int column=0;column<grades  [row].length;column++)
    {
                    if(grades[row][column]>highGrade)
                        highGrade=grades[row][column];
                }
            }
            return highGrade;
        }
    }
```

运行代码，得到如图 7-9 所示的结果。

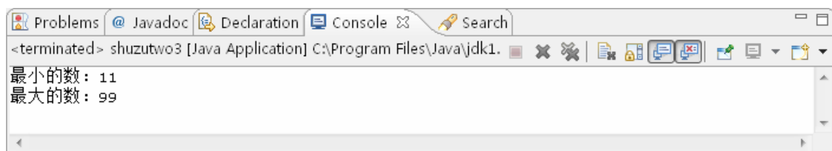


图 7-9　执行结果

多学一招

在上面的一个实例中，在数组中寻找数组的最大元素和最小元素是十分常用的操作，尤其是在公司办公的应用最为广泛，如查询本月工资情况、年龄等一些数组，都会求最大值和最小值。下面将用另一种方法计算二维数组的最大值和最小值，其代码见“光盘：源代码/第 7 章/shuzutwo4.java”：

```
    public class shuzutwo4
    {
            public static void main(String args[])
    {
            int maxage;
            int minage;
            //声明一个二维数组
            int[][] rage=new int[3][3];
            //赋值
            rage[0][0]=42;
            rage[0][1]=43;
            rage[0][2]=44;
```

```
            rage[1][0]=47;
            rage[1][1]=59;
            rage[1][2]=24;
            rage[2][0]=18;
            rage[2][1]=49;
            rage[2][2]=58;
    //高级应用
            maxage=rage[0][0];
            minage=rage[0][0];
            for(int i=0;i<rage.length;i++){
                for(int j=0;j<rage[0].length;j++){
                    if(maxage<rage[i][j]) maxage=rage[i][j];
                    if(minage>rage[i][j]) minage=rage[i][j];
                }
            }
            System.out.println("数组如下");
            for(int i=0;i<rage.length;i++)
            {
                for(int j=0;j<rage[0].length;j++)
                {
                    System.out.println("rage["+i+"]["+j+"]="+rage
[i][j]);
                }
            }
            //显示结果
            System.out.println("最大年龄"+maxage+",");
            System.out.println("最小年龄"+minage+",");
        }

    }
```

运行代码，得到如图 7-10 所示的结果。

```
Problems  @ Javadoc  Declaration  Console ☒  Search
<terminated> shuzutwo4 [Java Application] C:\Program Files\Java\jdk1.
数组如下
rage[0][0]=42
rage[0][1]=43
rage[0][2]=44
rage[1][0]=47
rage[1][1]=59
rage[1][2]=24
rage[2][0]=18
rage[2][1]=49
rage[2][2]=58
最大年龄59,
最小年龄18,
```

图 7-10　执行结果

7.3 多维数组

多维数组实际上就是二维数组和一维数组的升级版，在一些情况下，仅仅一维数组和二维数组仍然不能描述一种相同类型的数据，这就需要用到多维数组。在多维数组里，应用最为广泛的是三维数组，本节将主要讲解三维数组。

7.3.1 三维数组的声明

要声明三维数组，方法十分简单，它和一维数组、二维数组相似，其格式如下：

```
float a[][][];
char b[][][];
```

参数介绍如下：

- float：数组类型。
- a：数组名称。
- b：数组名称。

7.3.2 三维数组的创建

创建一个三维数组也是十分的简单，其格式如下：

```
int[][][] a=new int[2][2][3];
```

上面这个数组定义了一个三维数组是 2*2*3，读者可以将其想象成一个 2*3 二维数组即可。

7.3.3 三维数组的初始化

初始化多维数组十分简单，下面以一个三维数组为例进行讲解，其初始化多维数组的格式如下：

```
int[][][]a={
 //初始化三维数组
{{1,2,3}，{4,5,6}}
{{7,8,9},{10,11,12}}
}
```

上面代码定义了数组并且同时初始化元素值。

实例 30：使用三维数组

下面创建一个三维数组，然后将三维数组的元素打印出来，其代码见“光盘：源代码/第 7 章/shuzuduo1.java”：

```
public class shuzuduo1
{
   public static void main(String args[])
   {
```

```
                int[][][] a=new int[2][2][];
                a[0][0]=new int[2];
                a[0][0][0]=1;
                a[0][0][1]=2;

                a[0][1]=new int[2];
                a[0][1][0]=3;
                a[0][1][1]=4;

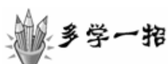
                a[1][0]=new int[2];
                a[1][0][0]=5;
                a[1][0][1]=6;

                a[1][1]=new int[2];
                a[1][1][0]=1;
                a[1][1][1]=1;
                        for(int i=0;i<a.length;i++)
                {
                    for(int j=0;j<a[0].length;j++)
                    {
                        for(int z=0;z<a[0][0].length;z++)
                        {
                            System.out.println("a["+i+"]["+j+"]["+z+"]="+a
[i][j][z]);
                        }
                    }
                }
            }
        }
```

运行代码，得到如图 7-11 所示的结果。

```
Problems  @ Javadoc  Declaration  Console  Search
<terminated> shuzuduo1 [Java Application] C:\Program Files\Java\jdk1.
a[0][0][0]=1
a[0][0][1]=2
a[0][1][0]=3
a[0][1][1]=4
a[1][0][0]=5
a[1][0][1]=6
a[1][1][0]=1
a[1][1][1]=1
```

图 7-11　执行结果



在上面的一个实例中，理解了一个简单的三维数组，下面再展示一段代码，以加深对多

维数组的操作。代码将产生一个随机数，定义数组然后将其打印，其代码见“光盘：源代码/第 7 章/shuzutwo4.java”：

```
public class shuzuduo2
{
   public static void main(String[] args)
   {
         int[][] ar1={
                     { 54, 2313, 54, 5641, 3 },
                     { 554, 46544, 13 },
               { 5454, 54 }
                     };
         for (int i=0; i < ar1.length; i++)
         {
         for (int j=0; j < ar1[i].length; j++)
         System.out.println(ar1[i][j]);
         }
         }
}
```

运行代码，得到如图 7-12 所示的结果。

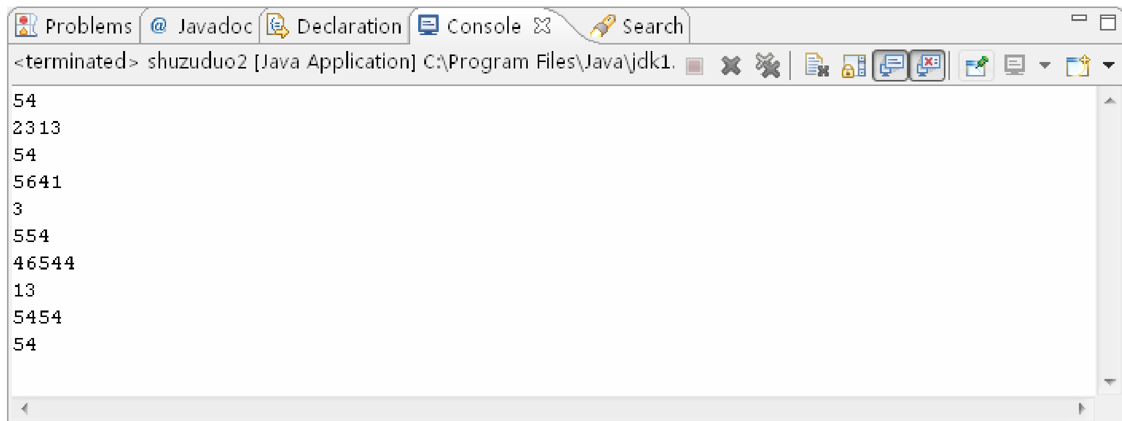


图 7-12　多维数组

7.4　对数组的操作

定义数组和初始化数组都是十分简单的，读者在学习的过程中，一定要掌握操作数组的技巧。数组是相同数据类型的一次集合，熟练操作它有很大的意义，在下面的讲解里，将讲解几种常用的操作数组的方法。

7.4.1　复制数组

在 Java 程序中，System 提供了一个特殊的方法 arraycopy()，用于实现数组之间的复

制，其格式如下：

```
System.arraycopy(arrayA,0,arrayB,0,a.length);
```

参数介绍如下：

- array A：来源数组名称。
- 0：来源数组起始位置。
- array B：目的数组名称。
- 0：目的数组起始位置。
- arrayA.length：复制来源数组元素的个数。

上面这个复制数组的功能有一定局限性，下面将这个方法改写，以强化它的功能，使它可以复制数组的任何元素，其格式如下：

```
System.arraycopy
(arrayA,2,arrayB,3,3);
```

参数介绍如下：

- array A：来源数组名称。
- 2：来源数组起始位置第 2 个元素。
- array B：目的数组名称。
- 3：目的数组起始位置第 3 个元素。
- 3：在来源数组第 2 个元素开始复制 3 个元素。

使用复制数组功能，需要明白 arraycopy()的 5 个参数，它们分别是来源数组、来源数组数据起始的位置、目的数组、目的数组数据起始位置、复制数组多少个数组元素。下面通过一个例子来讲解。

实例 31：实现一维数组的元素

下面通过一段代码讲解一维数组的元素复制，其代码见“光盘：源代码/第 7 章/shuzugong1.java”：

```
public class shuzugong1
 {
   public static void main(String[] args)
   {
         int X;
         int Y[]={ 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };
         int Z[]=new int[10];
         System.arraycopy(Y, 0, Y, 0, Y.length);
         for (X=0; X < Y.length; X++)
               System.out.print(Y[X] + " ");
         System.out.println();
   }
}
```

运行代码，得到如图 7-13 所示的结果。

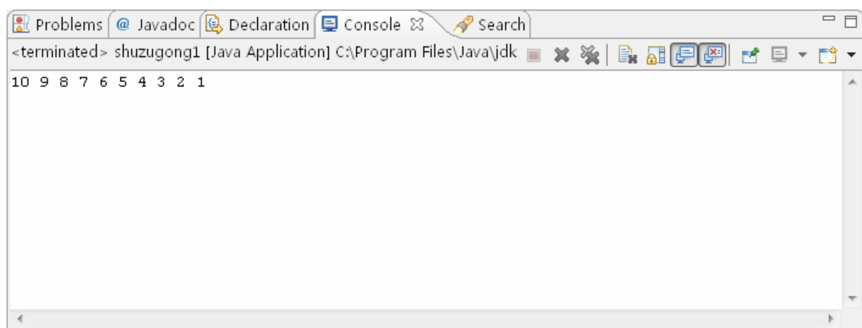


图 7-13　数组的复制

多学一招

在上面的一个实例中，讲解了复制数组元素的操作方法，下面再通过一段代码讲解另外一种复制数组元素的方法，其代码见“光盘：源代码/第 7 章/shuzugong2.java”：

```
public class shuzugong2
{
   public static void main(String[] args)
   {
//定义数组
         int a;
         int Aa[] = { 10,20,30,40,50,60,70,80,90,100 };
         int Bb[] = new int[10];
         System.arraycopy(Aa, 2, Bb, 2, 3);
         for (a = 0; a < Bb.length; a++)
               System.out.print(Bb[a] + " ");
         System.out.println();
   }
}
```

运行代码，得到如图 7-14 所示的结果。

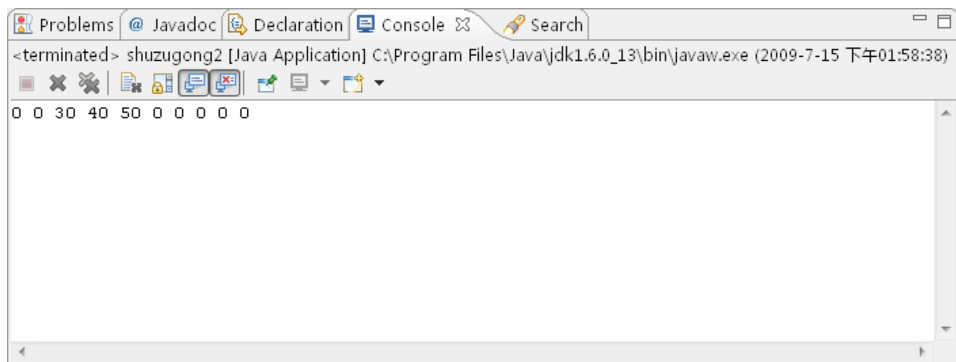


图 7-14　复制指定的元素

7.4.2 比较数组

比较数组就是比较两个数组是否相同，如果相同，则返回一个 true 值，如果不相同，则返回一个 false 值，其格式如下：

```
    Arrays.equalse(arrayA,arrayB);
```

参数介绍如下：

- arrayA：待比较数组名称。
- arrayB：待比较数组名称。

如果两个数组相同就会返回 true，如果两个数组不相同就会返回 false。

实例 32：将两个数组进行比较

下面将两个数组进行比较，数组相同返回一个 true 值，不同返回一个 flase 值，其代码见“光盘：源代码/第 7 章/shuzugong3.java”：

```
import java.util.Arrays;
public class shuzugong3
{
        public static void main(String[] args)
        {
             int[]a1={1,2,3,4,5,6,7,8,9,0};
             int[]a2=new int[9];
             System.out.println(Arrays.equals(a1, a2));
             int[]a3={1,2,3,4,5,6,7,8,9,0};
             System.out.println(Arrays.equals(a1, a3));
             int[]a4={1,2,3,4,5,6,7,8,9,5};
             System.out.println(Arrays.equals(a1, a4));
        }
   }
```

运行代码，得到如图 7-15 所示的结果。

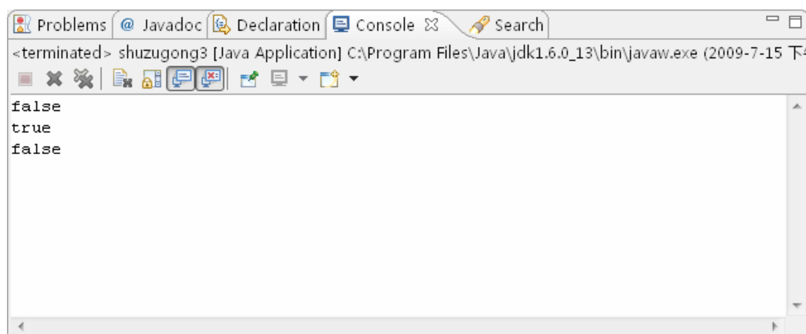
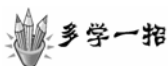


图 7-15　比较数组

提示：在对数组进行比较的时候，一定要在程序开头编写一句“import. java. util. Arrays;”否则程序会自动报错。这句程序的意思是插入软件包 Arrays，在后面的章节中会讲到。



比较数组的方法十分简单，用户也很容易判断两个数组是否相同。下面将展示一段代码，再次巩固对比较数组这个知识点的学习，其代码见“光盘：源代码/第 7 章/shuzugong4.java”：

```
import java.util.Arrays;
public class shuzugong4 {
   public static void main(String[] args)
   {
         int[]A1={1,2,3,4,5,6,7,8,9,0};
         int[]A2=new int[9];
         System.out.println(Arrays.equals(A1, A2));
         int[]A3={1,2,3,4,5,6,7,8,9,0};
         System.out.println(Arrays.equals(A1, A3));
         int[]A4={1,2,3,4,5,6,7,8,9,5};
         System.out.println(Arrays.equals(A1, A4));
         System.out.println(Arrays.equals(A2, A3));
         System.out.println(Arrays.equals(A2, A4));
   }
}
```

运行代码，得到如图 7-16 所示的结果。

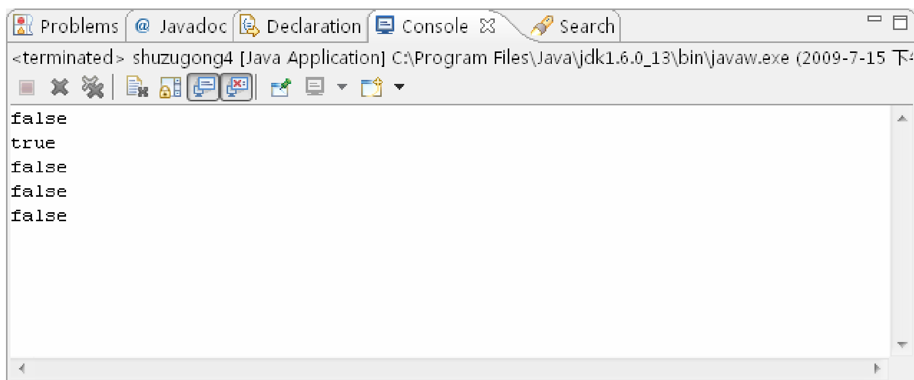


图 7-16　比较数组

7.4.3　搜索数组中的元素

搜索数组中元素的方法很简单，它的格式如下：

```
int i=binarySearch（a，”abcde”）;
```

参数介绍如下：

- ❑ a：搜索数组的名称。
- ❑ abcde：需要在数组中查找的内容。

代码 32：从程序的格式可以看到，使用它非常简单。下面通过一段代码进行讲解，其

代码见“光盘：源代码/第 7 章/shuzugong5.java”：

```
import java.util.Arrays;
import java.util.Comparator;
public class shuzugong5
{
        public static void main(String[] args)
        {

              int[] Aa={6,2,5,4,6,2,3};
              Arrays.sort(Aa);
              Arrays.binarySearch(Aa, 5);
              System.out.print("排序后的数组为：");
              for(int i=0;i<Aa.length;i++){
                  System.out.print(+Aa[i]+" ");
              }
              System.out.println();
                  int location=Arrays.binarySearch(Aa, 4);
System.out.println("查找 4 的位置是"+location+",Aa["+location+"]="
+Aa[location]);
                  }
   }
```

运行代码，得到如图 7-17 所示的结果。

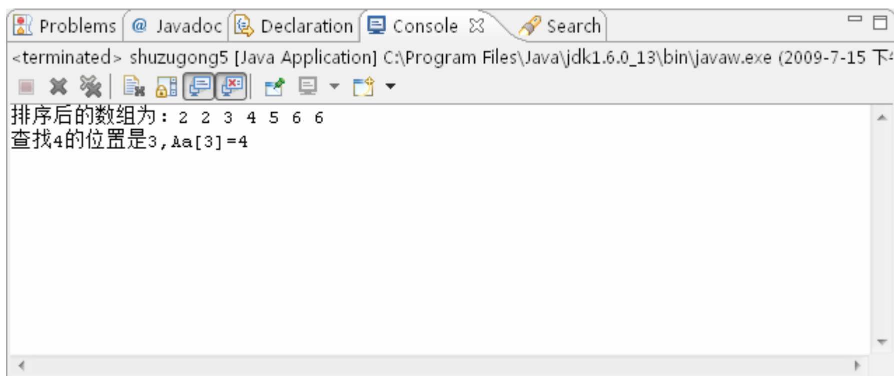


图 7-17　搜索数组

7.4.4　排序数组

在上面的学习中，每一个方法对应一个方法，排序也不例外，它也有自己专门的使用方法，对数组中进行排序的方法如下：

```
Arrays.sort(a);
```

其中 a 是排序数组名称。

代码 33：排序数组的使用方法十分简单，下面通过一段简单的代码进行讲解，其代码

见“光盘：源代码/第 7 章/shuzugong6.java”：

```
import java.util.Arrays;
public class shuzugong6
{
        public static void main(String[] args)
        {
            String []a=new String[] {"123","XYZ","ABCD","256"};
            Arrays.sort(a);
            System.out.println(Arrays.asList(a));
        }
  }
```

运行代码，得到如图 7-18 所示的程序。

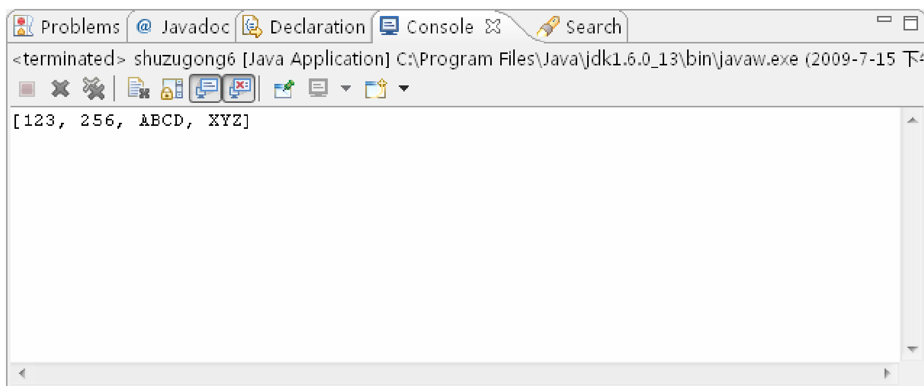


图 7-18　排序数组

7.4.5　填充数组

在 Java 程序设计中，有一个 fill()方法，它是用来填充数组的，这个方法有很大局限性，只能使用同一个数值进行填充。下面将展示一个新的填充数组的方法，其格式如下：

```
int a []=new int[10];
Arrays.fill(array,11);
```

其中 a 为填充数组的名称，这个程序的含义是将数值 11 填充到数组 a 中。

代码 34：下面通过一段代码讲解如何填充数组，其代码见“光盘：源代码/第 7 章/shuzugong 7.java”：

```
import java.util.Arrays;
public class shuzugong7 {
   public static void main(String[] args)
   {
            int size=0;
            if(args.length!=0)
```

```
                    size=Integer.parseInt(args[0]);
                int[]a1=new int[size];
                Arrays.fill(a1, 11);
                for(int i=0;i<a1.length;i++){
                    System.out.print("a1["+i+"]="+a1[i]+" ");
                }
                System.out.println();
            }

      }
```

若要运行这个程序，还需要如下的操作步骤：

1）选择“Run/Run Configurations”命令，如图 7-19 所示。

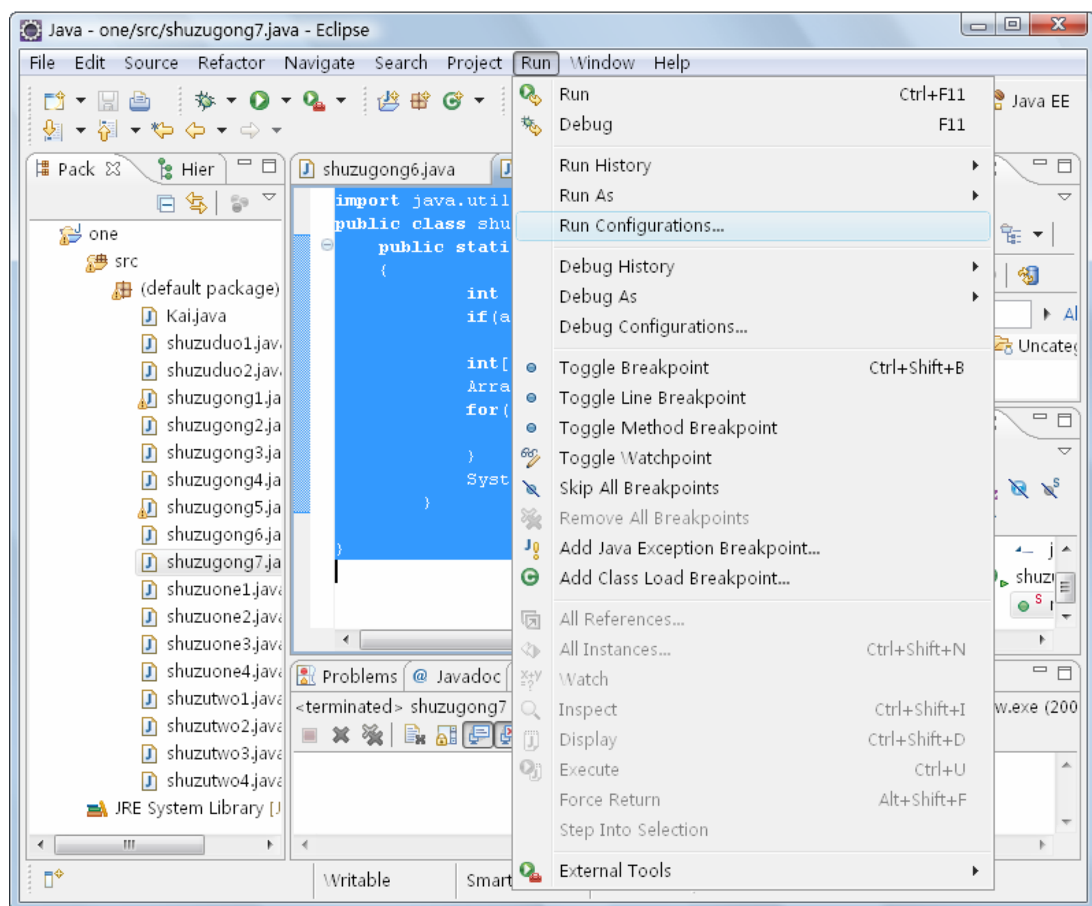


图 7-19　选择命令

2）打开“Run configurations”对话框，在对话框里不做任何设置，如图 7-20 所示。

3）单击打开右边的“Arguments”选项卡，在“Program arguments”中设置参数为“6”，如图 7-21 所示。

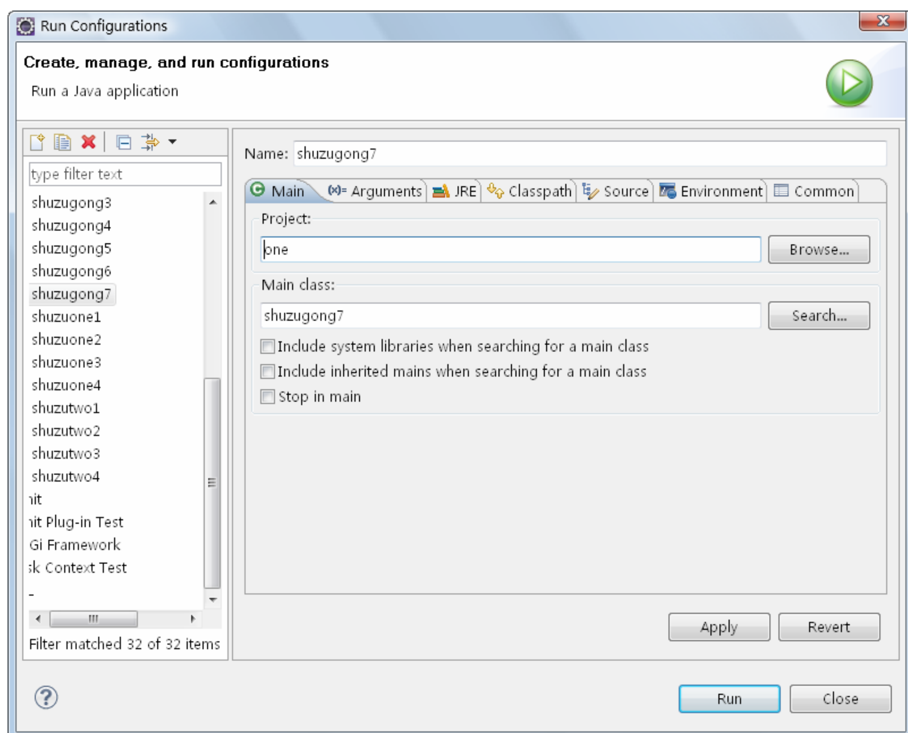


图 7-20　设置参数

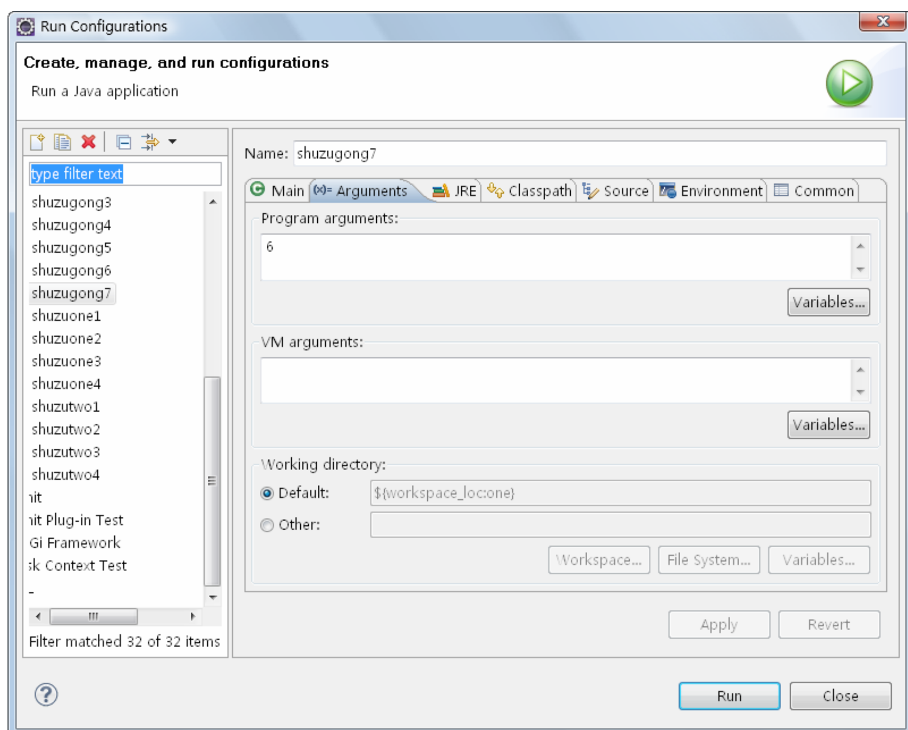


图 7-21　设置运行参数

4）设置完成后，单击“Run”按钮，将会得到如图 7-22 所示的结果。

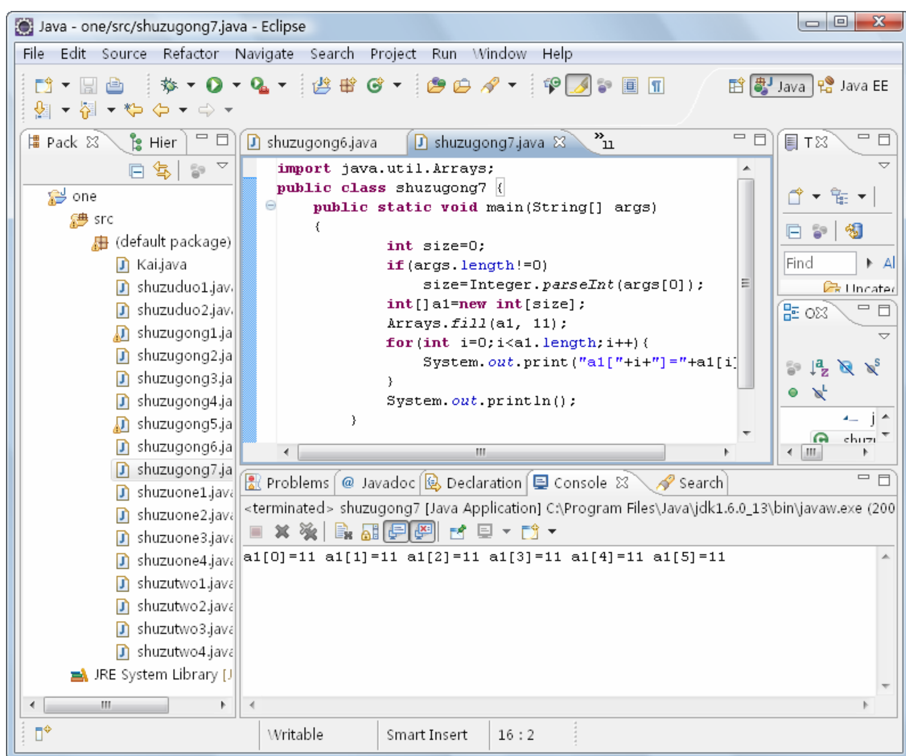


图 7-22　运行结果

提示：如果直接执行这个程序，将不会得到任何结果，看到的就是一片空白，这不是这个程序运行的结果，因为它缺少环境变量。

7.5　疑难问题解析

本章详细介绍了数组、对数组简单操作的基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：在操作数组时，为什么要在程序首行编写“import java.util.Arrays; ”语句？

解答：在 Java 程序里，默认情况下并不能操作数组，而是需要插入一个包才能操作数组，而“import java.util.Arrays;”主要就是用来插入操作数组的包，说明程序里需要这个包里的方法或者属性。

读者疑问：操作数组功能是不是只针对一维数组？这些方法能操作二维或则三维数组吗？

解答：操作数组的方法适用于任何数组，为了让大家更好理解，本章所讲解的操作数组的方法都是使用一维数组，为的是让读者快速理解。



职场点拨——客户沟通之道

在职场中的人有时会不可避免要去和客户沟通、相处，作为一名程序员当然也不会例外。当面对性格各异的客户时，怎样才能和它们好好相处，从而完成自己的任务呢？在此提出如下 4 条建议。

1）都说客随主便，在对方的“地盘”上我们要矜持。在客户单位时，我们要少谈论业务之外的问题，并且在谈论时要正儿八经，声音响亮，不能给外人一种偷偷摸摸的印象。并且要谨记，一切“潜规则”之类的细节绝不允许在客户的单位里谈，只要出现过一次这种事，客户便会对你非常反感。

2）一定要保证我们产品的质量，无论是 Web 项目还是桌面项目。不能让客户承担任何有可能的风险，如果客户感觉接受你的设计方案存在风险的话，客户也不敢收。

3）要跟客户说实话，不要自以为是地认为自己够聪明，其实客户并不傻。客户既然能代表公司谈合作，肯定是它们公司最优秀的计算机代表，并且已经接触了不只你一家软件公司。他甚至了解市面主流成本是多少钱。大多数客户非常反感在谈价格时说“低了多少钱，我连本都赚不回来”之类的话。一般负责项目洽谈的代表都是对计算机这一行业非常了解的，如果你想蒙骗客户，客户会直接拒绝与你合作。

4）在客户单位时，只要见到与客户同在一个办公室的任何人员，我们务必要谦逊。不管他是负责这个项目，还是不负责这个项目。你要对客户代表周围的人表现出极大的尊敬和热情，当然也要有一个度，只要让他身边的人认为你是值得信赖的人即可，不要让他们认为你是在巴结他们。

第 8 章　Java 面向对象

Java 是一门面向对象的语言，但是它是如何面向对象的呢？能准确说出答案的人很少。本章将讲解 Java 面向对象的一些知识与面向对象的一些特性。本章主要内容如下：

- 面向对象的概念。
- 类。
- 属性和方法的修饰符。
- 类和对象。
- 抽象类。
- 包。
- 职场点拨——打造一个团队。

2010 年 11 月 10 日，多云

凌晨时分，我看完一场英超联赛，有众多球星坐镇的豪门曼联竟然输给了一支实力差劲的弱旅。真是令人费解啊。



一问一答

小菜："昨晚看球赛了吗，曼联竟然输给了联赛排名倒数第一的球队!"

Wisdom："呵呵，昨晚的比赛我也看了，曼联的球星太单打独斗了，很少和队友做配合。足球是 11 人的集体运动，没有团队精神的球队输球很正常。"

小菜："足球场上团队合作很重要，那你说说 Java 体系中最重要的是什么？"

Wisdom："当然都很重要了，Java 应用范围太广了，想要精通任何一个领域都需要花费大量的精力。"

小菜："本章将要学的面向对象很重要吗？"

Wisdom："对初学者来说，面向对象思想很重要！你一定要了解高级语言的面向对象思想，因为它的推出是软件方式的一大进步，面向对象思想已经成为当前主流语言的核心思想。"

小菜："面向对象真有这么神奇吗？"

Wisdom："当然了，几乎所有的高级语言都是基于面向对象诞生的，它在 Java 体系中的地位犹如团体竞技项目中的团队合作。只有明白了面向对象思想，你才能真正认识到 Java 的强大。"

8.1　面向对象

Java 是一门纯面向对象的语言，所谓面向对象，就是将一切看做对象进行处理，在本节将详细讲解什么是面向对象。

8.1.1　面向对象的理念

什么是面向对象？面向对象的理念是什么？

面向对象（Object Oriented，OO）是当前整个计算机界关注的重点，它是 20 世纪 90 年代软件开发的主流方法。面向对象的概念和应用已超越了程序设计和软件开发，扩展到很宽的范围。如数据库系统、交互式界面、应用结构、应用平台、分布式系统、网络管理结构、CAD 技术、人工智能等领域。

谈到面向对象，这方面的文章比比皆是，但是能真正诠释对象的定义的却非常少。起初，“面向对象”是专指在程序设计中采用封装、继承、抽象等设计方法。可是，这个定义显然不再适合现在的情况。如今，面向对象的思想已经体现在软件开发的各个方面，比如，面向对象的分析（OOA，Object Oriented Analysis）、面向对象的设计（OOD，Object Oriented Design）以及人们经常说的面向对象的编程实现（OOP，Object Oriented Programming），等等。

8.1.2　面向对象的特点

面向对象编程更符合人们的思维模式，用它编写的程序更加健壮和强大。面向对象编程更有利于系统开发时责任的分工，能有效地组织和管理一些比较复杂的应用程序地开发。下面讲解面向对象程序的特点，其介绍如下。

- **对象唯一性：**每个对象都有自身唯一的标识代号，通过这种标识代号，可找到相应的对象。在对象的整个生命期中，它的标识都不改变，不同的对象不能有相同的标识代号。
- **抽象性：**指将具有一致的数据结构(属性)和行为(操作)的对象抽象成类。一个类就是这样一种抽象，它反映了与应用有关的重要性质，而忽略其他一些无关内容。任何类的划分都是主观的，但必须与具体的应用相联系。
- **继承性：**指子类自动共享父类数据结构和方法的机制，这是类之间的一种关系。在定义和实现一个类的时候，可以在一个已经存在的类的基础之上进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。如图 8-1 所示，时尚手机类继承了手机类，然后添加了 photograph( )方法和 playmp4 方法。

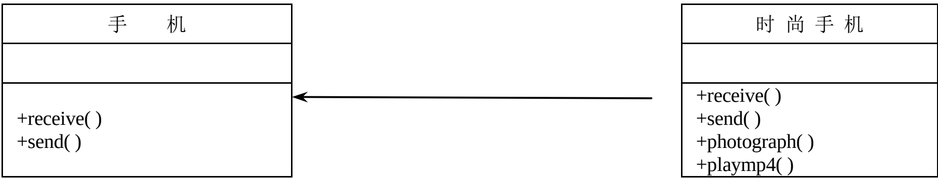


图 8-1　使用 UML 类图描述的手机类与时尚手机类

- **多态性**：指多种表现形式，具体来说，可以用"一个对外接口，多个内在实现方法"表示。举一个例子，计算机中的堆栈可以存储各种格式的数据，包括整型、浮点型和字符型。不管存储的是何种数据，堆栈的算法实现是一样的。
- **封装性**：指把被描述的客观世界事物的静态属性及相关操作的对象封装，而不必公开对象的属性和操作。

8.2 面向对象的第一特征——类

不管什么语言，只要它是面向对象的语言，就一定有类，如 C++、C#、网络编程语言 PHP 等都有类。所谓类，就是将相同属性的东西放在一起，如人，如猩猩，都可以是一个类。在 Java 中，每一个源程序至少都会有一个类，下面将详细讲解什么是类以及为什么源程序中会出现类。

8.2.1 如何编写一个类

在面向对象的程序中，首先要将一个对象看做一个类，假定人是对象，任何一个人都是一个对象，而类是一个大概念，因此这些对象具有一定的属性和方法。

代码 35：下面定义一个人的类，这是具有一定特效的一类事物，而 tom 则是类的一个对象实例，其代码如下：

```
class person {
//人具有 age 属性
   int age;
//人具有 name 属性
   String name;
//人具有 shut 方法
   void shut(){
         System.out.println("My name is"+name);
   }
public static void main(String args[]){
//类及类属性和方法的使用
person Tom=new person();
Tom.age=27;
Tom.name="TOM";
Tom.shut();
}
```

在一个类中只有属性和方法两种东西，属性是描述对象的，而方法是让对象实现功能的。

8.2.2 特殊的方法——构造方法

当一个类创建对象的时候，Java 会调用该类的构造方法。构造方法的命名必须与类名一致，否则将会造成编译错误。构造方法之所以特殊在于无论是否定义构造方法，所有的类都

会自动定义构造方法，倘若用户定义了构造方法，以用户定义为准。如果没有定义，就调用默认的构造方法，其格式如下：

```
[构造方法修饰符]方法名([参数列表])
{
方法体
}
```

实例 33：在创建类中创建一个构造方法

下面创建一个类，在里面创建一个构造方法，其代码见“光盘：源代码/第 8 章/leione.java”：

```
public class leione
 {
   String gname;
   int    gid;
   float  gprice;
   public void print(){
         System.out.println(" 商 品 名 "+gname+", 产 品 序 列 号 "+gid+", 价 格 是
"+gprice);
   }
   public static void main(String args[]){
         leione book1=new leione();
         book1.gname="湖南烤鸭";
         book1.gid=10005601;
         book1.gprice=78.0F;
         book1.print();
     leione book2=new leione();
         book2.gname="重庆火锅底料";
         book2.gid=1222002;
         book2.gprice=35;
         book2.print();
               }
}
```

运行代码，得到如图 8-2 所示的结果。

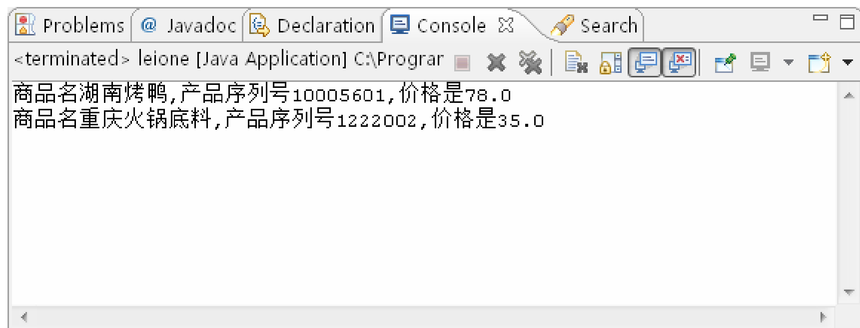
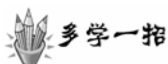


图 8-2　执行结果



在上面的实例中，学习了构造方法，下面将对上面这段代码进行修改，其代码见“光盘：源代码/第 8 章/leione1.java”：

```
public class leione1
{
   String gname;
   int  gid;
   float  gprice;
        public void print()
   {
        System.out.println("书名"+gname+",ISBN 序列号"+gid+",价格是"+gprice);
   }
        public leione1(String name,int id,float price)
   {
        gname=name;
        gid=id;
        gprice=price;
   }
   public static void main(String args[])
   {
        leione1 book1=new leione1("《大国时代》",987343334,45.0F);
        book1.print();

        leione1 book2=new leione1("《麻辣医师》",984545445,29.8F);
        book2.print();

        leione1 book3=new leione1("《中国不高心》",983432234,29.8F);
        book3.print();
              }
}
```

运行代码，得到如图 8-3 所示的结果。

提示：在多学一招里，相对实例来说代码更为简单，赋值没有那么麻烦，只需挨着顺序编写就可以了。倘若将“book3.print();”这个语句改写成“book2.print();”，试问执行结局会怎样？为什么呢？

修改后的结果如图 8-4 所示。

原因很好理解，虽然为 book3 赋了值，但是并没有要求显示它的值。而 book2 在之前赋了值，而且改写 book2.print()；使它再次调用了 print 方法，于是造成 book2 的值显示了两遍，初学者一定要加深理解。

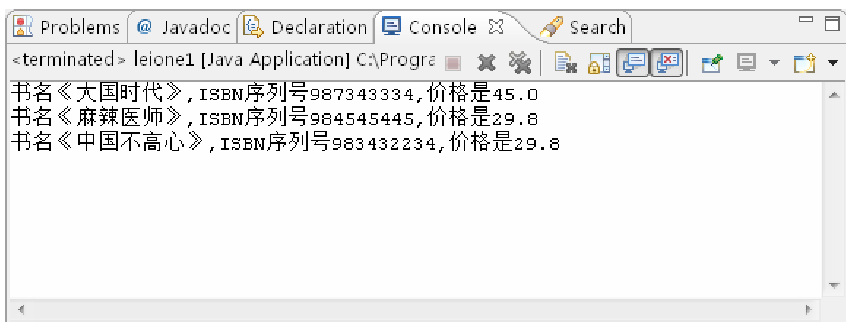


图 8-3　构造方法

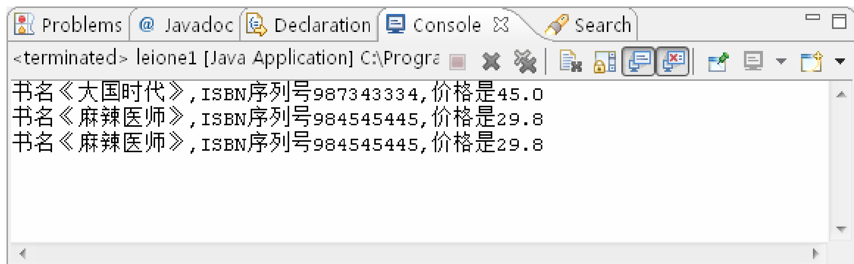


图 8-4　执行结果

8.2.3　一般的方法

在 Java 程序中，方法是用来实现类的某种功能的，一般的方法主要由声明和方法体两部分组成，其格式如下：

```
[修饰符]<方法返回值类型><方法名>([参数]){
方法体
}
```

其参数介绍如下：

- **修饰符**：这是一个可选参数，用于指定方法的权限，可选值为 public、protected 和 private。
- **方法返回值类型**：必选参数，用于指定方法的返回值类型，如果该方法没有返回值，可以使用关键字 void 进行修饰，方法返回的数据类型可以是符合 Java 的一切数据类型。
- **方法名**：必选参数，用户指定方法的名称，方法名必须是合法的 Java 标识符。
- **参数列表**：可选参数，用户指定方法所需要的参数，当存在多个参数时，各参数之间应使用逗号隔开，方法的参数类型可以是符合 Java 的一切数据类型。
- **方法体**：方法实现部分。

读者朋友实际上在前面的章节中已经多次接触过方法，其中“public static void main(String args[]){}”这句代码使用得最多。从方法结构来看，它就是一个方法，下面将展示一段代码，以加深对方法的理解，代码如下：

```
//定义一个无返回值的方法
public void cheng()
{
System.out.println(“我已经长大了”);
//…
}
//定义一个有返回值的方法
public int Da()
{
int a=100;
return a;
}
```

8.3　属性和方法的修饰符

在前面的章节中，用户了解了类和属性的概念，在 Java 程序设计里，为了严格控制其访问权限，引进了修饰符这一概念。在本节中，将对修饰符进行详细讲解。

8.3.1　public 修饰符

在 Java 程序设计里，其属性和方法定义为 public 类型，那么属性和方法所在的类及其子类、同一个包中的类、不同包中的类都可以访问这些属性和方法，下面是一个关于 public 修饰符的例子。

实例 34：使用 public 的属性和方法

下面创建一个类，在里面创建 public 的属性和方法，其代码见“光盘：源代码/第 8 章/Leitwo1.java”：

```
public class Leitwo1
 {
   public int a;
   public void print()
   {
         System.out.println("a 的值为"+a);
   }
}
class textone
{
   public static void main(String args[])
   {
         Leitwo1 aa=new Leitwo1();
         aa.a=4478;
         aa.print();
   }
}
```

运行代码，得到如图 8-5 所示的结果。

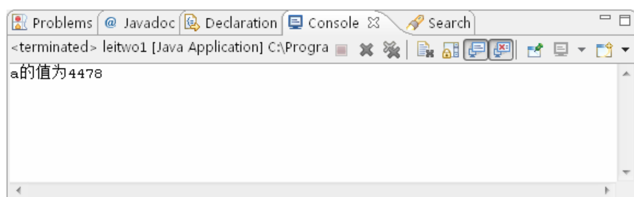


图 8-5　public 修饰符

多学一招

在上面的实例中，textone 类可以随意访问 Leitwo1 的方法和属性，让读者朋友对 public 有更进一步的认识。下面展示一段代码，巩固 public 修饰符的知识，其代码见“光盘：源代码/第 8 章/Leitwo2.java”：

```
public class Leitwo2
{
   public int Dong;
   public void print()
   {
         System.out.println("Dong 的值为"+Dong);
   }
}
class PublicTwo
{
   public static void main(String args[])
   {
         Leitwo2 aa=new Leitwo2();
         aa.Dong=234343;
         aa.print();
   }
}
```

运行代码，得到如图 8-6 所示的结果。

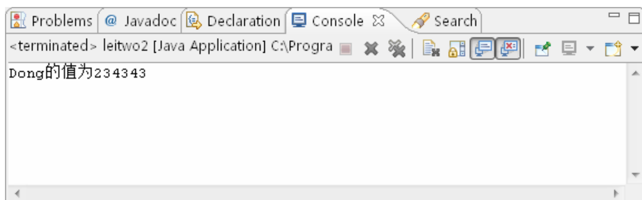


图 8-6　public 修饰符

8.3.2　private 修饰符

在 Java 程序里，倘若将属性和方法定义为 private 类型，那么该属性和方法只能在自己的类中被访问。

代码 36： 下面将展示一段代码，其代码见“光盘：源代码/第 8 章/Leitwo3.java”：

```
public class Leitwo3
{
   private String uname;
   private int uid;
         public String getuname()
   {
                     return uname;
   }
         private int getuid()
   {
         return uid;
   }
         public Leitwo3(String uname,int uid)
   {
         this.uname=uname;
         this.uid=uid;
   }
         public static void main(String args[])
   {
               Leitwo3 PrivateUse1=new Leitwo3("梁笑笑",21002);
               Leitwo3 PrivateUse2=new Leitwo3("秦奋",61002);
         String a1=PrivateUse1.getuname();
         System.out.println("姓名："+a1);
         int a2=PrivateUse1.getuid();
         System.out.println("学号："+a2);

         String a3=PrivateUse2.getuname();
         System.out.println("姓名："+a3);
         int a4=PrivateUse2.getuid();
         System.out.println("学号："+a4);
               }
}
```

运行代码，得到如图 8-7 所示的结果。

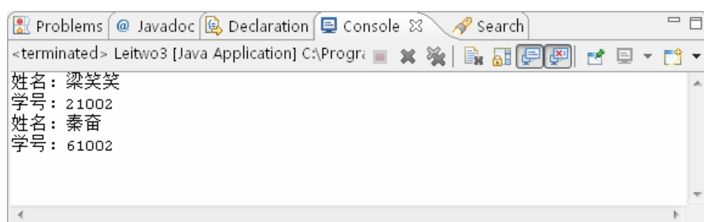


图 8-7　private 修饰符

8.3.3　protected 修饰符

在编写程序时，如果使用了修饰符“protected”修饰属性和方法，那么该属性和方法只能在自己的类和子类中访问。

代码 37：下面将展示一段代码，其代码见“光盘：源代码/第 8 章/Leitwo4.java”：

```
public class Leitwo4
{
   protected  int a;
   protected void print()
   {
         System.out.println("a="+a);
   }

    public static void main(String args[])
   {
         Leitwo4 a1=new Leitwo4();
         a1.a=2009;
         a1.print();
                     Leitwo4 a2=new Leitwo4();
         a2.a=2010;
         a2.print();
   }
}
```

运行代码，得到如图 8-8 所示的效果。

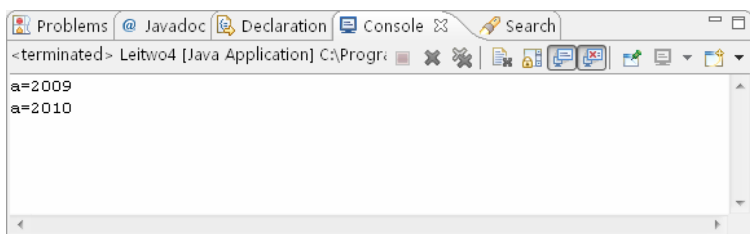


图 8-8　protected 修饰符

8.3.4　其他修饰符

前面三节讲解的三个修饰符是最常用的修饰符，除了这三个修饰符外，Java 程序设计里还有许多修饰符，下面对它们进行介绍：

- **默认修饰符**：如果没有指定访问控制修饰符，则表示使用默认修饰符，这时变量和方法只能在自己的类及该类同一个包下的类中访问。
- static：被 static 修饰的变量为静态变量，被 static 修饰的方法为静态方法。
- final：被 final 修饰的变量在程序在整个执行过程中最多赋一次值，所以它经常被定义为常量。

- ❑ transient：只能修饰非静态的变量。
- ❑ volatile：同 transient 一样，它只能修饰变量。
- ❑ abstract：被 abstract 修饰的成员称为抽象方法。
- ❑ synchronized：只能应用于方法，不能修饰类和变量。

实例 35：使用默认修饰符

下面创建一个类，并使用默认修饰符创建属性和方法，其代码见“光盘：源代码/第 8 章/leitwo5.java”：

```
public class leitwo5
{
   int a;
   int b;
   void print()
   {
         int c=a+b;
         System.out.println("a+b="+c);
   }
}
class UserOne1
{
   public static void main(String args[])
   {
         leitwo5 a1=new leitwo5();
         a1.a=2;
         a1.b=3;
         a1.print();
   }
}
```

运行代码，得到如图 8-9 所示的效果。

多学一招

在上面的程序中，属性全局变量和方法的访问权限修饰符都是默认的，因此类在 UserOne 中访问是用默认的方法 Print()。由此可见，变量和方法对于自己所在的类和默认的包（包的知识会在后面讲解）下的类都是可见的。下面是一段关于使用 static 修饰符的代码，其代码见“光盘：源代码/第 8 章/Leitwo6.java”：

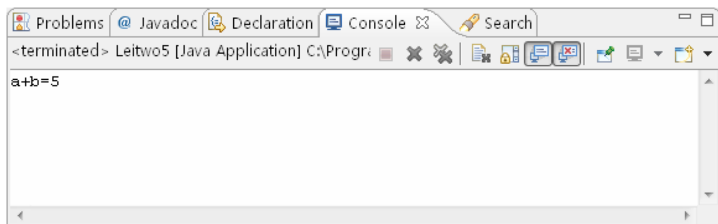


图 8-9　默认修饰符

```
public class Leitwo6
 {
   static int a;
   static int b;
         public static void print()
   {
         int c=a+b;
         System.out.println("a+b="+c);

   }
   }
class UseTwo1
{
   public static void main(String args[])
   {
         Leitwo6.a=10;
         Leitwo6.b=20;
         Leitwo6.print();
   }

}
```

运行代码，得到如图 8-10 所示的结果。

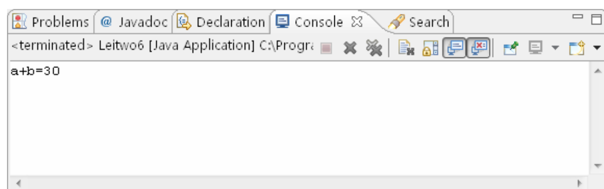


图 8-10 static 修饰符

8.4 this 的用法

前面已经讲解过变量，相信读者一定记得变量分为局部变量和全局变量。当局部变量和全局变量数据类型和名称都相同时，全局变量将会被隐藏，不能够使用。为了解决这样一个难题，Java 编程语言引用关键字 this 去访问全局变量，其格式如下：

```
this.成员变量名
this.成员方法名()
```

代码 38：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 8 章/leithree1.java”：

```
public class leithree1
{
```

```
        public String color="粉红色";//定义全局变量
        //定义一个方法
        public void hu()
        {
            String color="咖啡色";
        //定义局部变量
            System.out.print ("她的外套是"+color+"色的");
        //此处应用了局部变量
            System.out.print("她的外套是"+this.color+"色的");
        //此处应用了全局变量
        }
        }
```

如果直接运行这段代码将不会产生任何结果，这是因为没有编写 main 方法。但是 this 在 hu 方法里已经顺利访问了全局变量，如果用户要显示它，就需要编写一个 main 方法去调用 hu 方法。编写完成后，代码如下：

```
    public class leithree1
    {
        public String color="粉红色";//定义全局变量
        //定义一个方法
        public void hu()
        {
            String color="咖啡色";
        //定义局部变量
            System.out.println ("她的外套是"+color+"色的");
        //此处应用了局部变量
            System.out.println("她的外套是"+this.color+"色的");
        //此处应用了全局变量
        }
        public static void main(String args[])
        {

            leithree1 bb=new leithree1();
            bb.hu();
        }
    }
```

运行代码，得到如图 8-11 所示的结果。

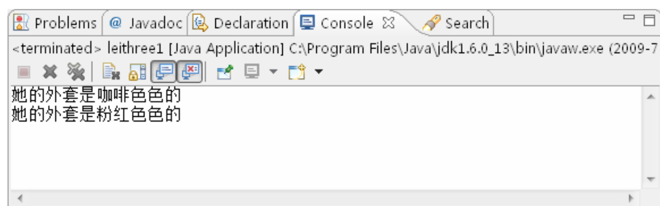


图 8-11　this 的使用

8.5　类和对象的使用

对象的使用，实际上就是引用对象的方法和变量，通过“.”可以实现对变量的访问和方法的调用。在 Java 程序中，方法和变量都有一定的访问权限，public、protected 和 private 等，通过一定的访问权限来允许或者限制其他对象的访问。

8.5.1　创建和使用对象

在 Java 程序中，一般是通过关键字 new 创建对象，计算机会自动为对象分配一个空间，然后访问变量和方法。不同的对象变量也是不同的，而方法则是由对象调用，下面通过一个实例进行讲解。

实例 36：在类中创建和使用对象

下面创建一个类，然后在类中创建并使用对象，其代码见“光盘：源代码/第 8 章/leidui1.java”：

```
public class leidui1
{
   int X=12;
   int Y=23;
         public void printFoo()
   {
         System.out.println("X="+X+",Y="+Y);
   }
   public static void main(String args[])
   {

         leidui1 Z=new leidui1();
         Z.X=41;
         Z.Y=75;
         Z.printFoo();

         leidui1 B=new leidui1();
         B.X=23;
         B.Y=38;
         B.printFoo();
   }
}
```

运行代码，得到如图 8-12 所示的结果。

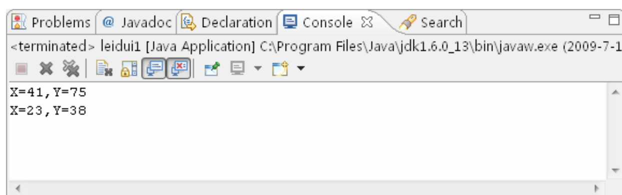
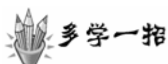


图 8-12　创建对象和使用对象



上面的实例中是创建对象和使用对象，下面将这段代码稍稍修改一下，其代码见“光盘：源代码/第 8 章/ leidui2.java”：

```
public class leidui2
{
   int X;
   int Y;
         public void printFoo()
   {
         System.out.println("X="+X+",Y="+Y);
   }
         public static void main(String args[])
   {
         leidui2  A=new leidui2 ();
         A.X=1;
         A.Y=2;
         A.printFoo();

         leidui2  B=new leidui2 ();
         B.X=4;
         B.Y=9;
         B.printFoo();
   }
}
```

运行代码，得到如图 8-13 所示的结果。

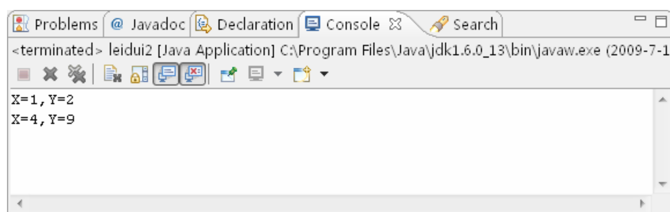


图 8-13　创建和使用对象

8.5.2　使用静态变量和静态方法

在前面的内容中已经讲过，变量和方法前面只要在使用修饰符 static，这个变量和方法就被称做静态变量和静态方法。访问静态变量和静态方法只需要类名，通过运算“.”即可以实现对变量的访问和对方法的调用。

实例 37：使用静态变量和静态方法

下面创建一个类，在里面定义了静态变量和静态方法对象，然后是访问静态变量的，其代码见“光盘：源代码/第 8 章/leijing1.java”：

```
public class leijing1
{
   static int X;
   static int Y;
         public void printJingTai()
   {
         System.out.println("X="+X+",Y="+Y);
   }
         public static void main(String args[])
   {
               leijing1 Aa=new leijing1();
         Aa.X=4;
         Aa.Y=5;
         leijing1.X=112;
         leijing1.Y=252;
         Aa.printJingTai();
         leijing1 Bb=new leijing1();
         Bb.X=3;
         Bb.Y=8;
         leijing1.X=131;
         leijing1.Y=272;
         Bb.printJingTai();
                     }
}
```

运行代码，得到如图 8-14 所示的结果。

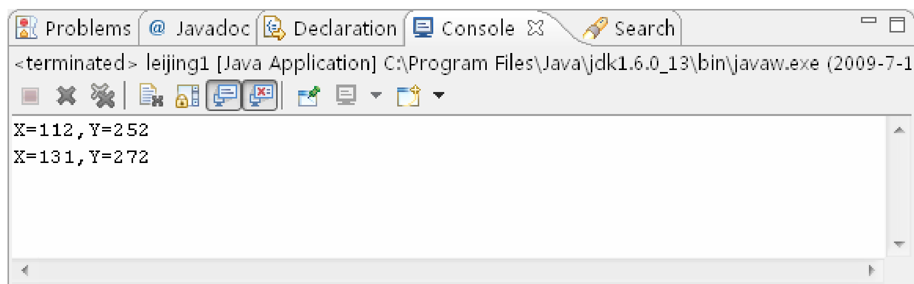
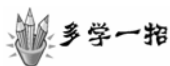


图 8-14　使用静态变量和方法



上面的代码讲解了如何用 new 运算符创建了一个对象，下面将展示一个关于对象调用静态方法的例子，其代码见“光盘：源代码/第 8 章/ leijing2.java”：

```
public class leijing2
{
```

```
        static int a,b;
        public static int getarea()
    {
        return a*b;
    }
        public static int getperi()
    {
        return 2*(a+b);
    }
        public static void main(String args[])
    {
                leijing2 a1=new leijing2();
                leijing2.a=124;
                leijing2.b=337;
        System.out.println("area="+leijing2.getarea());
        System.out.println("peri="+leijing2.getperi());
    }
    }
```

运行代码，得到如图 8-15 所示的结果。

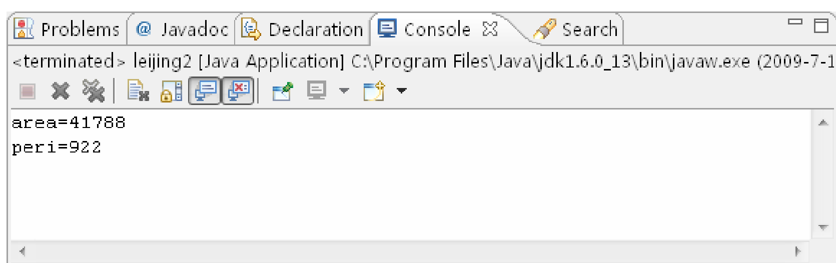


图 8-15　静态方法和变量

8.6　特殊的类—抽象类

明白了类后，再去理解抽象类就变得容易多了。在类之前加上关键字“abstract”就成了抽象类。有抽象类，就必定有抽象方法，抽象方法就是抽象类里的方法，下面将详细讲解抽象类。

8.6.1　创建抽象类

所谓抽象类就是只声明方法存在但不去实现它的类，抽象类不能实例化，也就是不能创建对象。在定义抽象类时，要在关键字 class 前加上关键字 abstract，其格式如下：

```
    abstract class 类名
    {
```

```
    类体
    }
```

实例 38：使用抽象类

下面创建一个抽象类，然后通过几个代码去实现它，其代码见“光盘：源代码/第 8 章/”：首先新建一个名为 Fruit 抽象类，其代码如下：

```
    public abstract class Fruit {
       //定义抽象类
       public String color;                          //定义颜色变量
       //定义构造方法
       public Fruit()
    {
             color="红色";                           //对变量 color 进行初始化
       }
       //定义抽象方法
       public abstract void harvest();               //收获的方法
    }
```

抽象类是不会具体实现的，如果不实现，那么这个类将不会有任何意义，所以需要新建一个类，来继承这个抽象类（继承的特性将在下一章讲解），其代码如下：

```
    public class pingguo extends Fruit
    {
       public void harvest()
       {
                   System.out.println("苹果已经收获！");
             }
    }
```

然后新建一个 juzi 的类，其代码如下：

```
    public class Juzi
    {
       public void harvest()
       {
             System.out.println("橘子已经收获！");
       }
    }
```

再新建一个 zong 的类，其代码如下：

```
    public class zong
     {
       public static void main(String[] args)
        {
                   System.out.println("调用苹果类的 harvest()方法的结果：");
                   pingguo pingguo=new pingguo();
                   pingguo.harvest();
```

```
                System.out.println("调用橘子类的 harvest()方法的结果：");
                Juzi orange=new Juzi();
                orange.harvest();
            }
    }
```

运行代码，将会得到如图 8-16 所示的结果。

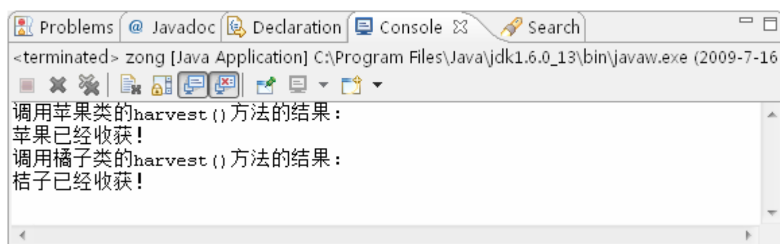


图 8-16　创建抽象类

8.6.2　抽象类的规则

抽象类最大的规则，就是一个抽象类必须有一个抽象方法。

代码 39：下面通过一段代码讲解抽象方法，其代码见“光盘：源代码/第 8 章/leichou.java”：

```
    abstract class Cou
     {
       int a1;
       int b1;
            Cou(int a,int b)
       {
            a1=a;
            b1=b;
       }
       abstract int mathtext();
    }

    class Cou1 extends Cou
    {
       Cou1(int a,int b)
       {
            super(a,b);
       }

       int mathtext()
       {
            return a1+b1;
       }
    }
```

```
class Cou2 extends Cou
{
   Cou2(int a,int b)
   {
         super(a,b);
   }
   int mathtext()
   {
         return a1-b1;
   }
}
public class leichou
{
   public static void main(String args[])
   {
         Cou1 abs1=new Cou1(3,2);
         Cou2 abs2=new Cou2(4,2);
         Cou abs;
         abs=abs1;
         System.out.println("加过后，它的值是"+abs.mathtext());
         abs=abs2;
         System.out.println("除过后， 它的值是"+abs.mathtext());
   }
}
```

运行代码，得到如图 8-17 所示的结果。

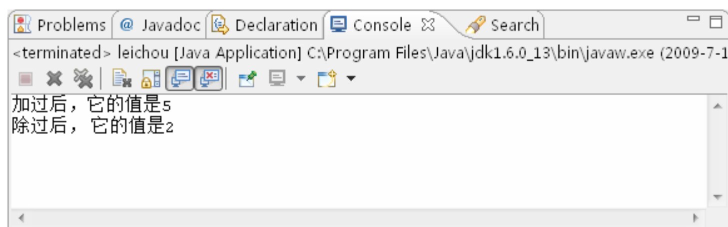


图 8-17　抽象类的规则

8.7　软件包

在开发程序时，有一些类或者方法是不能直接使用的，而是需要通过插入包的方法来调用这些类和方法。在本节中，将详细讲解如何定以包和插入软件包呢，以及包的含义。

8.7.1　定义软件包

定义软件包十分简单，用户只需在 Java 源程序中的第一句添加一段代码即可，其格式如下：

```
package 包名;
```

package 声明了多程序中的类属于哪个包，在一个包中可以包含多个程序。在 Java 程序中，用户还可以创建多层次的包，其格式如下：

```
package 包名 1[.包名 2[.包名 3]];
```

通过下面的代码即可创建一个多层次的包：

```
package China.CQ;
public class UseFirst
{
   public static void main(String args[])
   {
         System.out.println("这个程序定义了一个包");
   }
}
```

运行代码，得到如图 8-18 所示的结果。

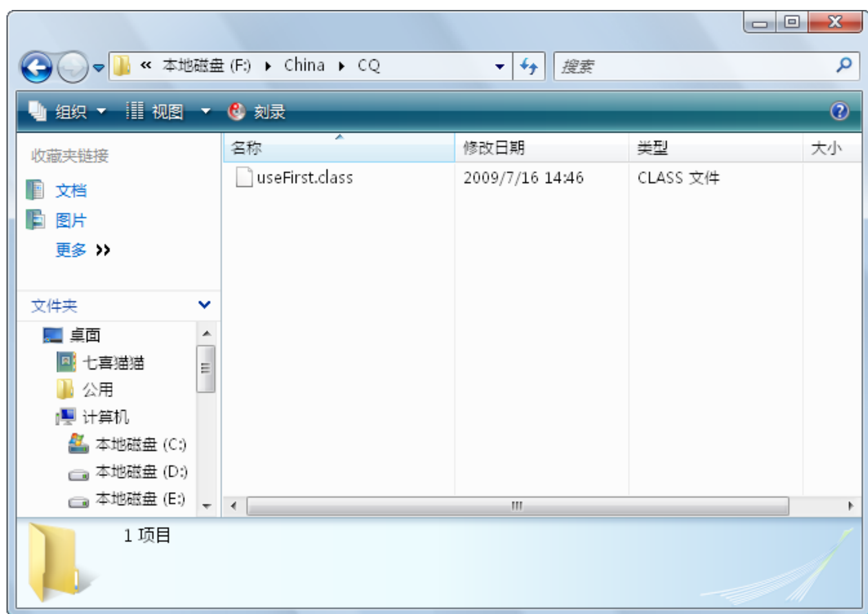


图 8-18　多层次软件包

提示：定义软件包实际上就是新建文件夹，并将编译后的文件放在新建的文件夹中。

8.7.2　在 Eclipse 中定义软件包

在 Eclipse 中定义软件包同样十分简单，具体操作如下：

1）使用鼠标选中项目并单击鼠标右键，在弹出的快捷菜单中选择“New/Package”命令，如图 8-19 所示。

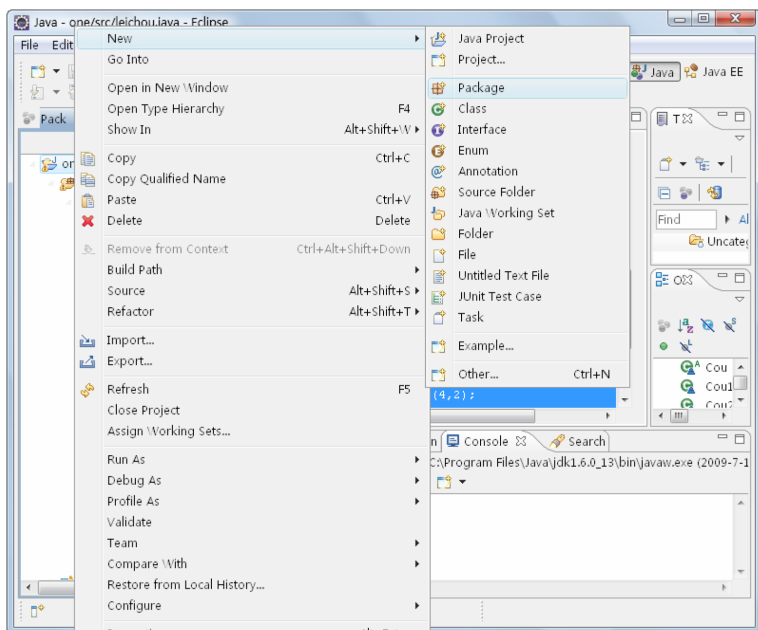


图 8-19　定义软件包

2）在弹出的窗口中，输入需要建立的软件包名，如果需要建立多级包，只需要用“.”符号隔开即可，如图 8-20 所示。

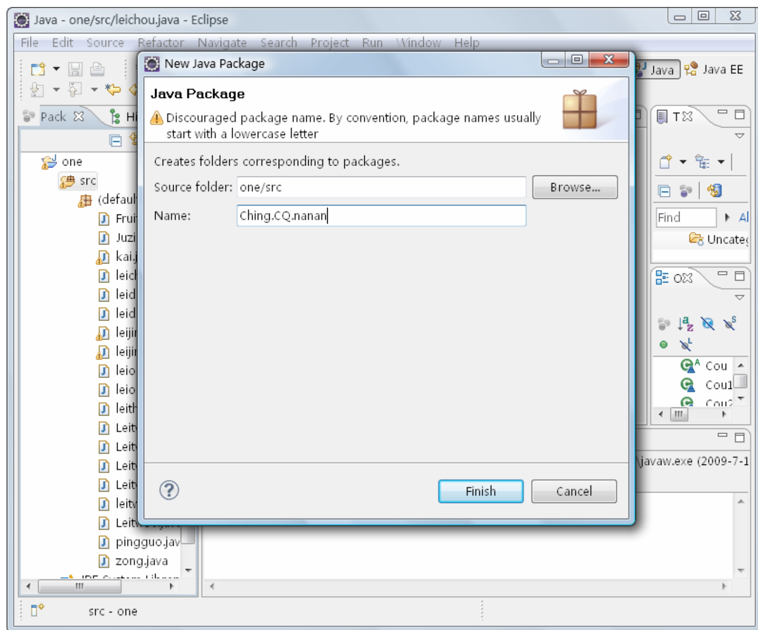


图 8-20　新建软件包

3）单击“Finish”按钮并建立源代码。之后选中新建的包，单击鼠标右键，在弹出的快捷菜单中选择“New/Class”命令，打开新的窗口，然后输入一个类名，如 Student，如图 8-21 所示。

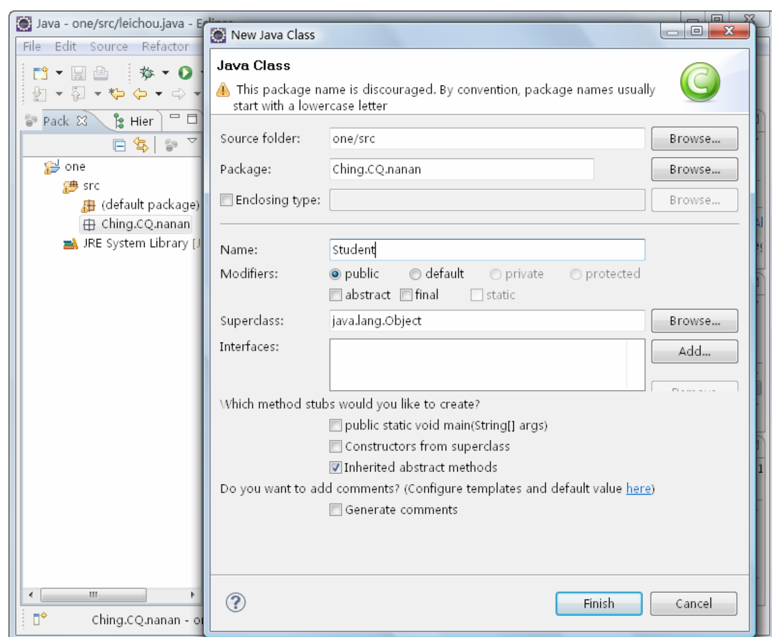


图 8-21　创建类

4）单击“Finish”按钮，这个类将会自动添加软件包名，如图 8-22 所示。

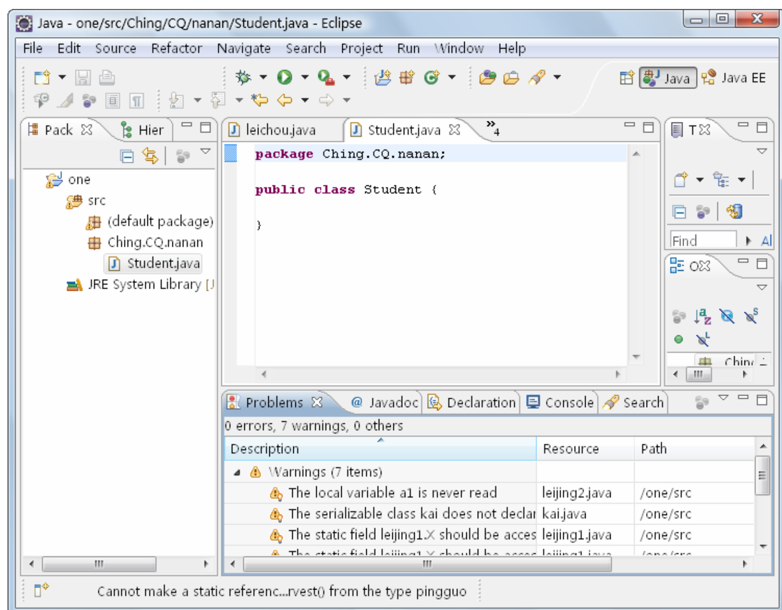


图 8-22　新建的包

提示：在 Eclipse 里，依上述步骤新建的 Java 程序会自动添加软件包。

8.7.3　在程序中插入软件包

插入软件包十分简单，需要使用 import 语句来插入所需要的类。在“数组”一章中，用

户对插入软件包这个概念有了初步了解，其格式如下：

```
import 包名1[.包名2…].(类名1*);
```

参数介绍如下：

- 包名 1：一级包。
- 包名 2：二级包。
- 类名：是需要导入的类的类名。也可使用*号，表示将导入这个包中所有的类。

实例 39：在类中插入一些特定的包

下面新建一个类，插入一些特定的包让这个类实现一些功能，其代码见“光盘：源代码/第 8 章/leibao.java”：

```
import java.util.*;
import java.awt.*;
import java.util.Date;
public class leibao
{
   int a;
   int b;
         public void print()
   {
         System.out.println("a="+a+",b="+b);
   }
     }
class BaoTwo
{
   public static void main(String args[])
   {
         leibao a1=new leibao();
         a1.a=121;
         a1.b=232;
         a1.print();
   }
}
```

运行代码，得到如图 8-23 所示的结果。

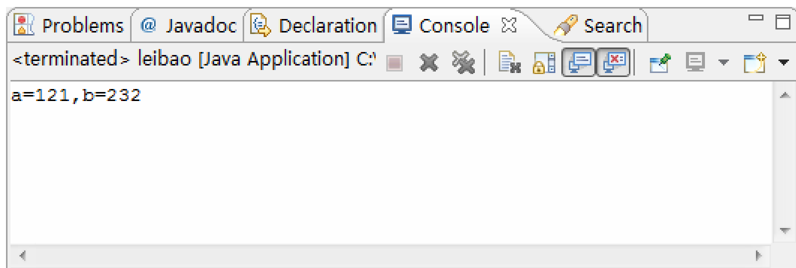
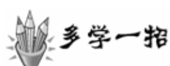


图 8-23　插入软件包



读者朋友可能发现，导入软件包实际上就是在类前写代码。在上面的程序中，插入了正确的包，倘若在一个程序中，插入了不同包中的相同类，而在程序中有这个类的方法，这时编译这段代码会出现什么情况呢？下面通过代码进行讲解，其代码见“光盘：源代码/第 8 章/leibao2.java”：

```
import java.awt.*;
import java.util.*;
import java.sql.*;
import java.io.*;
public class leibao2
{
   void kk()
   {
         System.out.println("导入语句");
   }

   public static void main(String args[])
   {
         Date a1=new Date();
                System.out.println(a1.getYear()+"年"+a1.getMonth()  +"月"+a1.getDate()+"日");
   }
}
```

运行代码，就会发现错误提示，如图 8-24 所示。

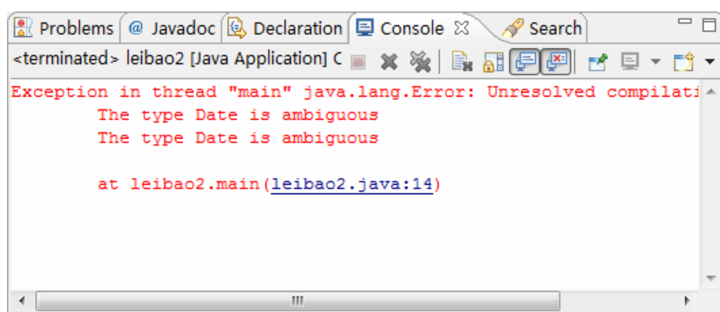


图 8-24　编译错误

提示：这个错误是类型 Date 有歧义。

要想正确运行上面的代码，解决的方法十分简单，只需要在 Date 类前加上软件包的全名，以告诉系统在访问的是哪一个包的 Date 类即可。修改后的代码如下：

```
import java.awt.*;
import java.util.*;
```

```
        import java.sql.*;
        import java.io.*;
        public class leibao2
        {
           void kk()
           {
                 System.out.println("导入语句");
           }

           public static void main(String args[])
           {
                 java.util.Date a1=new java.util.Date();
                        System.out.println(a1.getYear()+"年"+a1.getMonth() +"月
"+a1.getDate()+"日");
           }
        }
```

运行代码，得到如图 8-25 所示的结果。

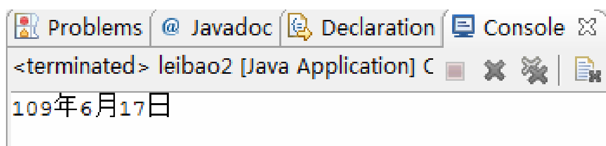


图 8-25　插入软件包

8.8　疑难问题解析

本章详细介绍了面向对象的概念、属性、方法、修饰符的基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：面向对象该怎么理解？构造方法是做什么的？

解答：面向对象是编程的一种思路，就是将用户需求看做对象，然后对对象进行合理分层，构建相对独立的业务模块。构造方法就是类初始化的时候调用的方法。

读者疑问：听说抽象类是一种很特殊的类，那么在什么情况下才用抽象类呢？

解答：在做一个工程的时候，有很多重复的工作需要不同的类完成，这个时候，用户就可以使用抽象类定义所有工程都可能使用的方法，由其他的类继承这个类，快速调用类中的方法完成任务。



职场点拨——打造一个团队

足球要依靠团队的力量，不能靠某一个球星赢得比赛，职场也一样。在企业中的你应

该如何培养和建立自己的团队素养？如何融入团队？在团队中发挥作用并获得团队及个人的成功，这对于程序员来说是个不小的挑战。现在的软件开发已经不再是以前那个个人英雄主义时代，现在更多的情况都需要以团队的形式进行系统设计和开发，团队精神也变得越来越重要。那么到底什么是团队精神呢，我觉得它包括了如下 4 个特点。

（1）荣辱与共

作为一个团队中的成员，就要把整个团队的荣辱放在第一位，这似乎是集体主义精神的体现，与当前更为流行的以个人为中心的思想有些背道而驰，但是，只有把整个团队的利益放在首位，团队才能够发展和进步。而团队的发展和进步必定会给其中的每个成员带来好处。

（2）交流分享

交流不但在团队中重要，在任何工作中都是非常重要的。人和人之间需要充分交流后才能够更好地工作，团队之中每个成员之间都应该充分交流，否则会在信息的传达过程中出现理解上的偏差。如果第一阶段工程（需求分析、概要设计）的负责人不和实施阶段工程（详细设计、编码、测试）的人员充分交流，那么很可能会做出一个客户不满意的产品。

（3）精诚协作

作为一个团队中的成员，也一定要牢记这四个字。想要达到精诚协作，首先就要摒弃“事不关己，高高挂起”的思想。尽管有些工作不是我们份内的，但是既然都属于团队的事情，我们就有责任尽自己所能去做好工作。有人会说，做得多，错就多，帮别人修改了程序，当这个程序出问题的时候，就会怪罪到自己的头上。这种情况的确存在，笔者也曾遇到过很多次，但是我更珍惜的是在这个过程中和其他团队成员的交流以及所学习到的知识。任何事物都不可能是完美的，都具有两面性。而且这样做非常有利于形成真正意义上的团队，在出现问题的时候，我们帮助过别人，当我们自己出现问题的时候，也就会有人帮我们。

（4）尊重理解

每个人都有自己的长处，也都有自己的短处，我们每个人只能尽量取长补短，谁也不能保证做到完美。生活中有很多其他的因素会对工作造成影响，当发现别人犯错的时候，我们应该理解，并且需要以对事不对人的态度去解决问题。例如测试人员发现开发人员开发的程序中有很多缺陷，这时不应该去指责，而是应该记录下来，然后和开发人员一起分析，提醒他以后不要再犯类似的错误。

第 9 章　类

类，简单来讲，就是将相同属性的元素放在一起，物以类聚，人以群分，折射的就是这个道理。类也是面向对象最有力的佐证，那么面向对象到底有什么特性呢？在本章将会详细讲解。本章主要内容如下：

- 类的继承。
- 重写和重载。
- 接口。
- 职场点拨——模块化设计的重要性。

2010 年 X 月 XX 日，晴

武侠小说是精美的成年人童话。我今天看了一本精彩的武侠小说，书中的主角有个神秘的箱子，能够根据不同的对手迅速组装成不同的武器，从而迅速制敌。我也很想拥有一个如此神奇的箱子啊！



一问一答

小菜：“刚刚看了一本很精彩的武侠小说，书中有一个神奇的箱子，可以组装成各种强大的武器，我很希望拥有那个神奇的箱子。”

Wisdom：“呵呵，武侠毕竟是虚构的，你还是好好学习吧！”

小菜：“我有个问题想问你，我发现市场上很多关于模块的书籍，模块和本章将要讲解的类有什么关系吗？”

Wisdom：“两者的原理是一样的，一个类是为了描述一个对象而定义的，而一个模块则是为了实现一个功能而编写的。模块设计的好处多多，它的原理和小说中神奇的箱子类似，能够为了满足某个功能而设计一段代码。当需要实现这个功能时，直接调用此模块即可。”

9.1　类的继承

类的继承，就是从已经定义的类中派生出一个新类，这就称为继承。继承是面向对象最重要的特征。本节将详细讲解继承。

9.1.1　父类和子类

继承是面向对象的机制，利用继承可以创建一个公共类，这个类具有多个项目的共同属

性。一些具体的类继承该类，同时拥有该类的属性和其自身特有的属性。继承的方法十分简单，其格式如下：

```
<修饰符>class<子类名>extends<父类名>
{
[<成员变量定义>]…
[<方法的定义>]…
}
```

人们常把子类称为父类的直接子类，把父类称为子类的直接超类，假如类 A 继承了类 B 的子类，必须符合下面的要求：

- 存在另一类 C，类 C 是类 B 的子类，类 A 是类 C 的子类，那么可以判断出类 A 是类 B 的子类。
- 在 Java 程序设计中，一个类只能有一个父类，即 extends 关键字前只能有一个类，它不支持多重继承。

实例 40：使用父类和子类

下面新建两个类，让其中一个类继承另一个类，其代码见“光盘：源代码/第 9 章/Jione1.java”如下：

```
class jitwo
{
 String name;
 int age;
 long number;
 jitwo(long number,String name,int age)
 {
    System.out.println("姓名 "+name);
    System.out.println("年龄" +age);
    System.out.println("手机号码 " +number);
 }
}

class super2b extends jitwo
{
 super2b(long number,String name,int age,boolean b)
 {
    super(number,name,age);
    System.out.println("喜欢运动？"+b);
 }
}
public class Jione1
{
 public static void main(String args[])
```

```
    {
        super2b abc=new super2b(15881,"陈佳瑶",18,true);
    }
}
```

运行代码，得到如图 9-1 所示的结果。

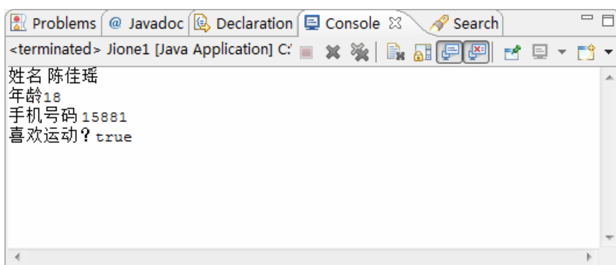
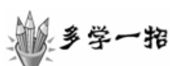


图 9-1　类的继承



在上面的一个实例中，super2b 类继承父类的属性以及方法。下面再展示一段代码，深入讲解类的继承，其代码见“光盘：源代码/第 9 章/Jitwo1.java”：

```
class Fuone{
 String name="张正义";
 int age=28;
 long number=12324343;
}
class Funew extends Fuone
{
 String name;
 int age;
 long number;
 void a0()
 {
    name="张城";
    age=19;
    number=23433438;
    System.out.println("姓名:  "+name);
    System.out.println("年龄:  "+age);
    System.out.println("电话:  "+number);
    name=super.name;
    age=super.age;
    number=super.number;
    System.out.println("姓名:  "+name);
    System.out.println("年龄:  "+age);
```

```
        System.out.println("电话：  "+number);
     }
    }
    public class Jitwo1
    {
     public static void main(String args[])
     {
        Funew abc=new Funew();
        abc.a0();
     }
    }
```

运行代码，得到如图 9-2 所示的结果。

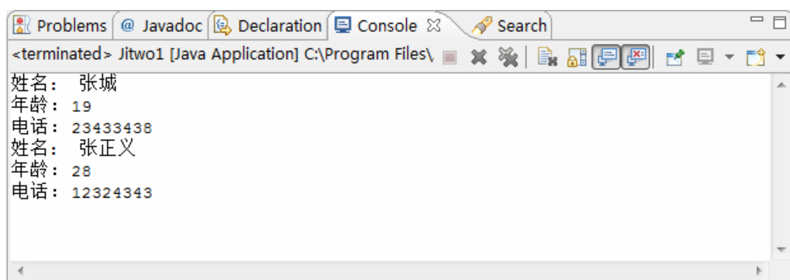


图 9-2　访问父类的属性

9.1.2　调用父类的构造方法

在类中，构造方法是比较重要的方法，子类要访问构造方法十分简单，格式如下：

```
    Super(参数);
```

实例 41：用子类去访问父类的构造方法

下面新建一个父类和子类，然后让子类去访问父类的构造方法，其代码见“光盘：源代码/第 9 章/ Newgou.java”：

```
    public class Newgou
    {
        String bname;
        int    bid;
        int    bprice;
                Newgou()
        {
            bname="乱石穿空";
            bid=322221;
            bprice=42;
        }
```

```
          Newgou(Newgou a)
    {
        bname=a.bname;
        bid=a.bid;
        bprice=a.bprice;
    }
          Newgou(String name,int id,int price)
    {
        bname=name;
        bid=id;
        bprice=price;
          }
          void print()
    {
System.out.println("书名："+bname+"序号："+bid+"  价格："+bprice);
    }
 }
 class Newgou1 extends Newgou
 {
     String Newgou;
          Newgou1()
    {
        super();//调用父类的构造方法
        Newgou="作家出版社";
    }
          Newgou1( Newgou1 b)
    {
        super(b);//调用父类的构造方法
        Newgou=b.Newgou;
    }
          Newgou1(String x,int y,int z,String aa)
    {
        super(x,y,z);//调用父类的构造方法
        Newgou=aa;
    }
 }
 class text1
 {
      public static void main(String args[])
      {
          Newgou1 a1=new Newgou1();
          Newgou1 a2=new Newgou1("物种起源",343006,45,"中国新世界出版集团");
          Newgou a3=new Newgou(a2);
          System.out.println(a1.Newgou);
          a1.print();
          System.out.println(a2.Newgou);
```

```
            a2.print();
            a3.print();
        }
    }
```

运行代码，得到如图 9-3 所示的结果。

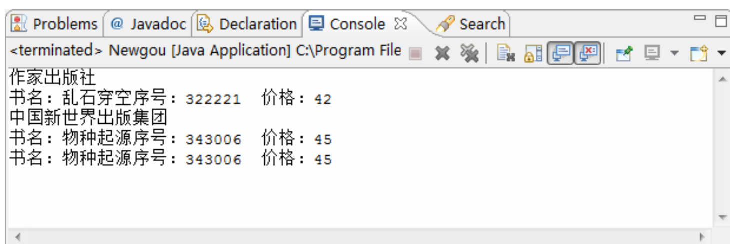
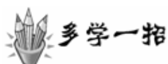


图 9-3　调用父类的构造方法



在上面的实例中讲解了用 super 关键字调用父类的构造方法，其实在类的继承中有这样一个特性，当子类明确不使用 super 关键字时，系统会自动调用父类中默认的构造方法。下面展示另一段代码，其代码见“光盘：源代码/第 9 章/ Newmo.java”如下：

```
class moone1
{
 moone1()
 {
     System.out.println("父类");
 }
}
class moone2 extends moone1
{
  moone2()
 {
     System.out.println("Nei1 继承父类 Nei");
 }
}
class Newmo extends moone2
{
 Newmo()
 {
     System.out.println("Nei2 继承父类 Nei1");
 }

 public static void main(String args[])
 {
     new moone2();
```

```
    }
}
```

运行代码，得到如图 9-4 所示的结果。

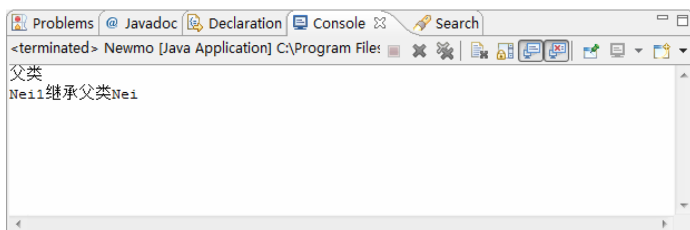


图 9-4　自动调用父类的构造方法

9.1.3　随意访问父类的属性和方法

访问父类的属性和方法的格式如下：

```
Super.[方法和全局变量];
```

实例 42：用子类去访问父类的属性

下面新建一个父类和子类，然后让子类去访问父类的属性，其代码见“光盘：源代码/第 9 章/ supertwo1.java”：

```
class supertwo1
{
 int a=11;
 int b=29;
}

class supertwo2 extends supertwo1
{
 int a=57;
 int b=89;

 supertwo2(int x,int y,int z,int q)
 {
     super.a=x;//调用父类被子类隐藏的变量
     super.b=y;
     a=z;
     b=q;
 }

 void print()
 {
     System.out.println(""+super.a);
     System.out.println(""+super.b);
```

```
        System.out.println(""+a);
        System.out.println(""+b);
        }
    }

    class text2
    {
     public static void main(String args[])
     {
        supertwo2 a1=new supertwo2(11,22,23,24);
        a1.print();
     }
    }
```

运行代码，得到如图 9-5 所示的效果。

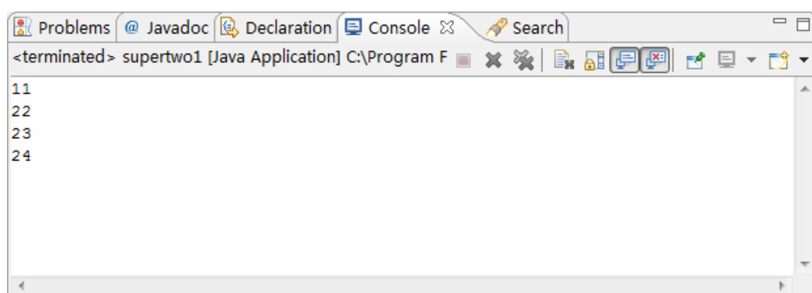


图 9-5　执行结果

多学一招

在上面的实例中，首先访问了父类的属性和方法，然后通过 text2 类进行实例化调用，但是上面这个代码，对于初学者来讲，其中有一点难以理解。下面再通过一段简单的程序代码，为读者加深理解，其代码见“光盘：源代码/第 9 章/ supertwo3.java”：

```
    class superkk1
    {
     String name="舒红";
     String adr="重庆渝北";
     int age=19;
     long number=67234343;
    }

    class superkk2 extends superkk1
    {
     String name;
     String adr;
     int age;
     long number;
```

```
void max()
{
    name="贾二刀";
    age=29;
    adr="重庆万州";
    number=123433438;
    System.out.println("姓名:  "+name);
    System.out.println("籍贯:  "+adr);
    System.out.println("年龄:  "+age);
    System.out.println("电话:  "+number);
    name=super.name;
    age=super.age;
    number=super.number;
    adr=super.adr;
    System.out.println("姓名:  "+name);
    System.out.println("籍贯:  "+adr);
    System.out.println("年龄:  "+age);
    System.out.println("电话:  "+number);
}
}

public class supertwo3
{
public static void main(String args[])
{
    superkk2 abc=new superkk2();
    abc.max();
}
}
```

运行代码，得到如图 9-6 所示结果。

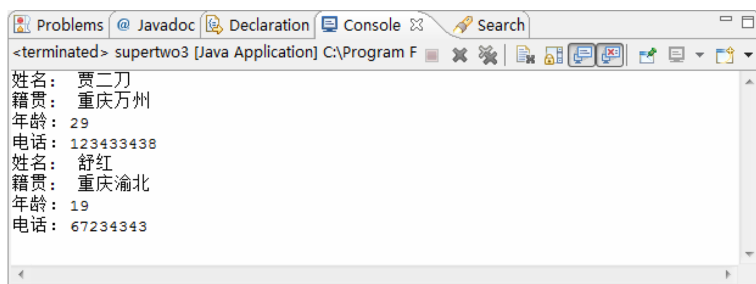


图 9-6　访问父类的属性和方法

9.1.4　多重次继承

多重次继承理解起来十分简单，如 B 类继承了 A 类，C 类继承了 B 类，这种情况就叫做 Java 的多重次继承。

代码 40：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 9 章/ Duolei.java”如下：

```
public class Duolei
{
 String bname;
 int    bid;
 int    bprice;
     Duolei()
 {
     bname="麻辣鸡翅";
     bid=14002;
     bprice=45;
 }

 Duolei(Duolei a)
 {
     bname=a.bname;
     bid=a.bid;
     bprice=a.bprice;
 }

 Duolei(String name,int id,int price)
 {
     bname=name;
     bid=id;
     bprice=price;

 }

 void print()
 {
     System.out.println("小吃名："+bname+"  序号："+bid+"  价格："+bprice);
 }
}

class Badder extends Duolei
{
 String badder;

 Badder()
 {
     super();
     badder="重庆小吃";
 }

 Badder( Badder b)
 {
```

```
        super(b);
        badder=b.badder;
    }

    Badder(String x,int y,int z,String aa)
    {
        super(x,y,z);
        badder=aa;
    }
}

class Factory extends Badder
{
    String factory;

    Factory()
    {
        super();
        factory="四川小吃";
    }

    Factory(Factory c)
    {
        super(c);
        factory=c.factory;
    }

    Factory(String x,int y,int z,String l,String n)
    {
        super(x,y,z,l);
        factory=n;
    }
}

class zero
{
    public static void main(String args[])
    {
        Factory a1=new Factory();
        Factory a2=new Factory("金华火腿",92099,25,"浙江小吃","金华小吃");
        Factory a3=new Factory(a2);
        System.out.println(a1.badder);
        System.out.println(a1.factory);
        a1.print();
        System.out.println(a2.badder);
        System.out.println(a2.factory);
        a2.print();
        a3.print();
```

```
        }
    }
```

运行代码，得到如图 9-7 所示的结果。

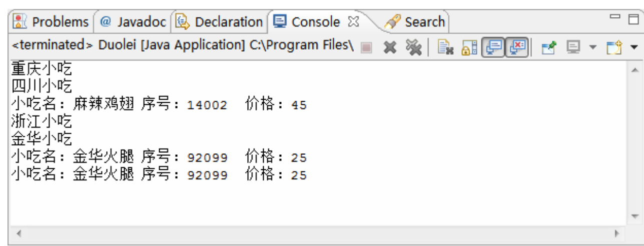


图 9-7　多重次继承

9.2　重写和重载

在面向对象中重写和重载十分重要，两者的名字听起来十分相似，其实是两种截然不同的概念，不过它们都能体现出 Java 的优越性。本章将详细讲解。

9.2.1　重写

重写是建立在类的继承基础之上的，它使 Java 语言结构变得丰富，对于初学者来说，重写是个难点，但是只要明白它的根本思想就变得很好理解。从一个类派生出一个新类，在父类中有方法，但是该方法不符合子类的要求，因此子类需要修改父类的方法，或自定义新的方法，这就是重写的思想。

（1）定义重写

重写实际上就是子类重新编写父类的方法，以达到自己的需要，这里通过一个实例讲解，如何定义方法的重写。

实例 43：定义重写

下面定义重写，让子类重写父类的方法，其代码见“光盘：源代码/第 9 章/chongxie.java”：

```
public class chongxie
{

 void print()
 {
     System.out.println("父类的方法");
 }
}

class Chongxieone extends chongxie
{
```

```
          void print()
          {
              System.out.println("子类，重写了父类的方法");
          }
         }
```

这段没有运行的结果，但是在父类中有 void print（）{}这个方法，子类重写了它，从而达到子类需要要求。


多学一招

在设计程序中是避免不了子类重写父类的。新定义的类，必然有新的特征，不然这个类也就没有意义了。在实例中，只是让读者明白如何重写，但并不具备实际的意义，下面给出一段完整的代码讲解重写的重要性，其代码见“光盘：源代码/第 9 章/ Cxie.java”：

```
         class Cxie
         {
          String sname;
          int    sid;
          int    snumber;
              void print()
          {
             System.out.println("公司名："+sname+" 序号："+sid+"公司人数："+snumber);
          }
              Cxie( String name,int id,int number)
          {
              sname=name;
              sid=id;
              snumber=number;
          }
         }

         class Cxietwo extends Cxie
         {
          String sadder;

          Cxietwo(String x,int y,int z,String aa)
          {
              super(x,y,z);
              sadder=aa;
          }

          void print()
          {
              System.out.println("学院/系别："+sname+"  序号："+sid+"  总人数：
"+snumber+"  地址："+sadder);
          }
         }
```

```
class gongsi
{
 public static void main(String args[])
 {
     Cxietwo a1=new Cxietwo("计算机系",21,2700,"西三楼");
     a1.print();
         }
}
```

运行代码，得到如图 9-8 所示的结果。

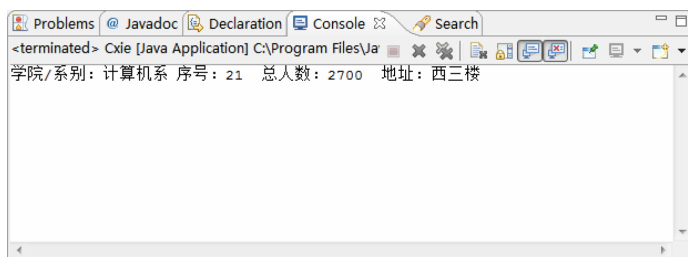


图 9-8　重写

（2）重写的规则

重写也有自己的规则，初学者需要牢记，父类中的方法并不是在任何情况下都可以重写的，当父类中的方法控制修饰符为 private 时，该方法只能被自己的类访问，不能被外部的类访问，是不能被子类重写的。如果定义父类的方法为 public，在子类定义为 private，程序运行时会报错，这一方面 Java 规定重写方法的权限不能比被重写的方法更严格，下面通过一个实例对重写的规则进行讲解。

实例 44：定义一个接口并编写一个抽象方法

在定义时讲解过，有的情况是不能重写父类方法的，下面通过一段代码讲解重写的规则，其代码见“光盘：源代码/第 9 章/ Cguize.java”：

```
class Cguize
{
 String sname;
 int   sid;
 int   snumber;

 public void print()
 {
     System.out.println("公司名："+sname+" 序号："+sid+" 公司人数："+snumber);
 }

 Cguize( String name,int id,int number)
 {
     sname=name;
     sid=id;
```

```
            snumber=number;
        }
       }

       class CguizeOne extends Cguize
       {
        String sadder;

        CguizeOne(String x,int y,int z,String aa)
        {
            super(x,y,z);
            sadder=aa;
        }

        private void print()//重写方法降低了访问权限
        {
            System.out.println("公司名为：\"+sname+"  序号：\"+sid+"  总人数：
"+snumber+"  公司地址："+sadder);
        }
       }

       class texttwo
       {
        public static void main(String args[])
        {
            CguizeOne a1=new CguizeOne("重庆金区公司",72221,7001,"渝南大道");
            a1.print();
                }
       }
```

运行代码，程序报错，如图 9-9 所示。

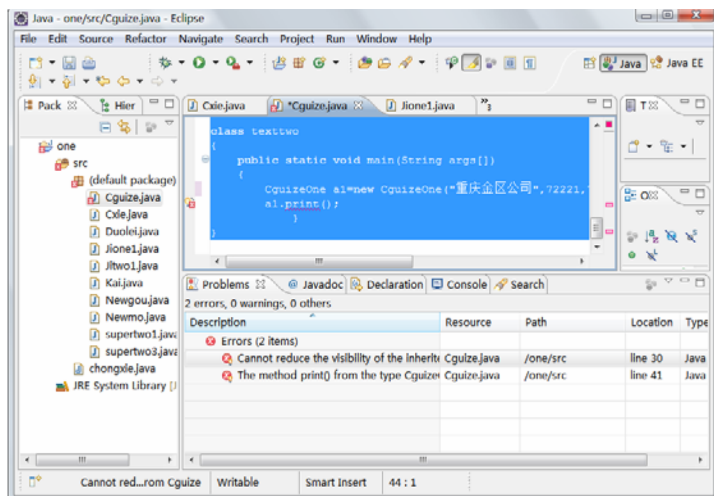
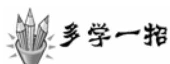


图 9-9　重写的规则



实例中的错误十分明显，是 print 方法出了问题。在子类重写方法时用了修饰符 private，而父类是 public，在 Java 程序中，private 要求明显比 public 更为严格，所以会得到错误提示。下面将讲解一段代码，让子类重写父类的方法，父类是抽象类，抽象方法是必须重写的方法。其代码见“光盘：源代码/第 9 章/ ChouDou.java”：

```
abstract class ChouDuo//定义的抽象类
{
 int a;
 int b;
 abstract void print();//定义了抽象方法
 ChouDuo(int x,int y)
 {
     a=x;
     b=y;
 }
}

class ChouDuo1 extends ChouDuo
{
 int c;
 ChouDuo1(int r1,int r2,int r3)
 {
     super(r1,r2);
     c=r3;
 }

 void print()//定义了属于自己的一个方法
 {
     System.out.println(a+b+c);
 }
}

class ChouDuo2 extends ChouDuo
{
 int d;
 ChouDuo2(int t1,int t2,int t3)
 {
     super(t1,t2);
     d=t3;
 }
 void print()//定义了属于自己的一个方法
 {
     System.out.println(a*b*d);
 }
```

```
        }

        class tian
        {
         public static void main(String args[])
         {
             ChouDuo1 a1=new ChouDuo1(11,72,23);
             ChouDuo2 a2=new ChouDuo2(13,23,33);
             a1.print();
             a2.print();
         }
        }
```

运行代码，得到如图 9-10 所示的结果。

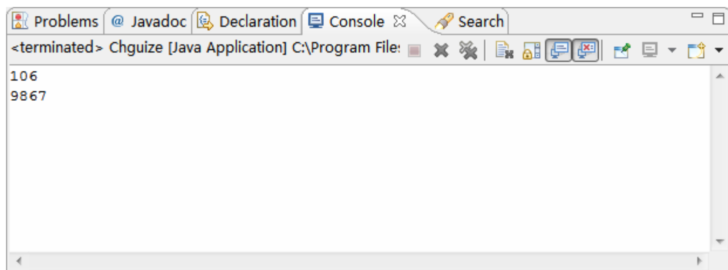


图 9-10　重写的规则

提示：方法重写支持多态性，多态性是面向对象程序设计的特性之一，一方面所有的子类继承了父类的所有元素。另一方面所有的子类根据自己需要，灵活定义自己的方法。在 Java 中，重写实现了多态性的“一种接口，多个方法”。

9.2.2　重载

重写和重载虽然不是同一个概念，但是它们也有相似之处，那就是他们都能体现 Java 的优越性。重载可以大大减少程序员的负担，开发者不需要记住那些复杂而难记的名称。本节将详细讲解方法的重载。

（1）定义重载

在 Java 程序中常使用重载来处理不同类型数据。所谓定义重载，就是在类中创建多个方法，它们具有相同的名字，但参数不同，调用方法通过传递给它们的不同参数个数和参数类型来决定具体使用哪个方法。下面通过一段代码讲解方法的重载，其代码见“光盘：源代码/第 9 章/ Czai.java”：

```
        public class Czai
        {
         String ename;
         int    age;

         void print()
```

```
    {
        System.out.println("姓名为: "+ename+" 年龄: "+age);
    }

    void print(String  a,int b)
    {
        System.out.println("姓名为: "+a+" 年龄: "+b);
    }

    void print(String a,int b,int c)
    {
        System.out.println("姓名为: "+a+" 年龄: "+b+" ID 号: "+c);
    }

    void print (String a,int b,double c)
    {
        System.out.println("姓名为: "+a+" 年龄: "+b+" ID 号: "+c);
    }
    }
   class textdir
   {
    public static void main(String args[])
    {
        Czai a1=new Czai();
        a1.ename="李中华";
        a1.age=28;
        a1.print();
        a1.print("成建国",27);
        a1.print("邓风",16,10025);
        a1.print("章雨",25,4125);
    }
   }
```

运行代码，其结果如图 9-11 所示。

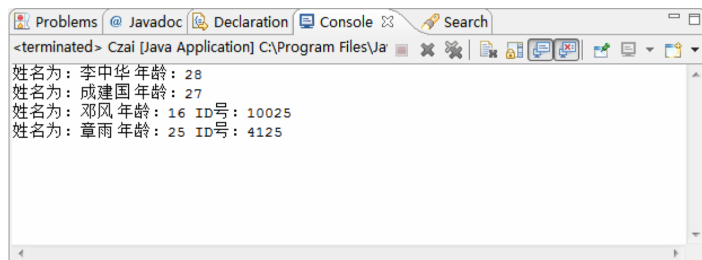


图 9-11　方法的重载

（2）重载的规则

重载的规则很简单，即参数决定重载方法的调用。当调用重载方法时，确定要调用那个

参数是基于其参数表的。如果是 int 参数调用该方法，则调用其自带的 int 方法，如果是 double 参数调用该方法，则调用其自带的 double 方法。

代码 41：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 9 章/ Ccand.java”：

```
public class Ccand
{
 int a;
 int b;

 int subtration()
 {
    return a-b;
 }

 int subtration(int a,int b)
 {
    return a-b;
 }

 int subtration(int a,int b,int c)
 {
    return a-b-c;
 }

 double subtration(double a,double b, double c)
 {
    return a-b-c;
 }

}

class textd
{
 public static void main(String args[])
 {
    Ccand a1=new Ccand();
    a1.a=3;
    a1.b=1;
    System.out.println(a1.subtration(13,21));
    System.out.println(a1.subtration(26,33,15));
    System.out.println(a1.subtration(29,34,32));
 }
```

运行代码，得到如图 9-12 所示的结果。

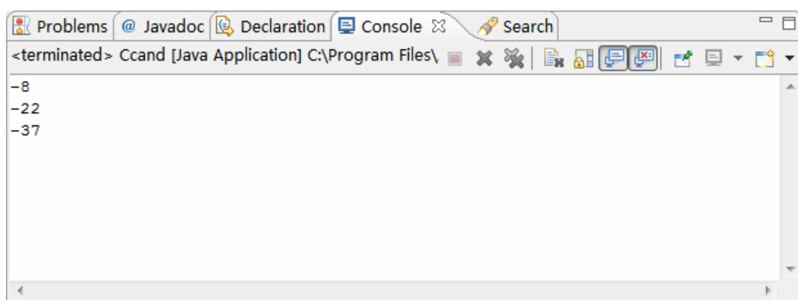


图 9-12　重载的规则

9.2.3　重写与重载联合使用

在同一段代码中有可能会同时出现重写和重载，从前面的知识可以知道，重写是指在子类中重新编写父类继承的方法，以让方法具有新的功能。而重载是在一个类中同方法名不同参数的方法，要明确两者针对的对象不同。

代码 42：下面通过一段代码讲解重写和重载，其代码见“光盘：源代码/第 9 章/Cfang.java”：

```
public class Cfang
{
 int a=101;
 int b=902;

 int print()
 {
     return a+b;
 }
 int print(int a,int b)
 {
     return a+b;
 }
}

class Cfang1 extends Cfang
{

 int print ()
 {
     return a;
 }

 double print(int a,double b)
 {
     return a+b;
 }
```

```
}

class textyo
{
 public static void main(String args[])
 {
     Cfang a1=new Cfang();
     Cfang1 a2=new Cfang1();
     a1.a=1;
     a1.b=2;
     System.out.println(a1.print());
     System.out.println(a1.print(13,22));

     a2.a=4;
     a2.b=5;
     System.out.println(a2.print());
     System.out.println(a2.print(33,22));
 }
}
```

运行代码，得到如图 9-13 所示的结果。

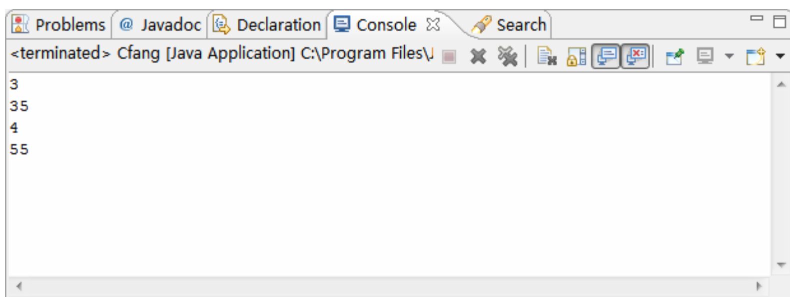


图 9-13　重载的规则

9.3　接口

在 Java 程序设计中，有一种元素和类的特性十分相似，这个元素就是接口。和类一样，在接口中也有方法，且接口可以派生出新的类，下面将详细讲解接口和接口的实现方法。

9.3.1　定义接口

接口的方法和抽象类中一样是抽象的，也就是说接口本身不具有实现的功能，但它可以被类实现，不论它指定要做什么，具体怎么做，一旦定义了接口，任何类都可以将它实现。接口与类不同，一个类只可以继承一个类，但是一个类可以实现多个接口，这在编写程序时，解决了一个类不能具备多个方面特征的难题。创建接口十分简单，格式如下：

```
[public] interface<接口名>{
[<常量>]
[<抽象方法>]
}
```

其参数介绍如下：

- public：接口的修饰符只能是 public，因为只有这样接口才能被任何包中的接口或类访问。
- interface：接口的关键字。
- 接口名：它的定义法则和类名一样。
- 常量：在接口中不能声明变量，因为接口要具备三个特征，公共性、静态的和最终的。

实例 45：定义一个接口并编写一个抽象方法

下面将定义一个接口，并编写一个抽象方法，其代码见“光盘：源代码/第 9 章/Newjie.java”：

```
interface JieKou1
{
 void print(int a);
 void print1(int b);
}

class JieKou implements JieKou1
{
 public void print(int a)
 {
     System.out.println("a="+a);
 }

 public void print1(int b)
 {
     System.out.println("b="+b);
 }
}

class Newjie
{
 public static void main(String args[])
 {
     JieKou a1=new JieKou();
     a1.print(1);
     a1.print1(2);
 }
}
```

运行代码，得到如图 9-14 所示的结果。

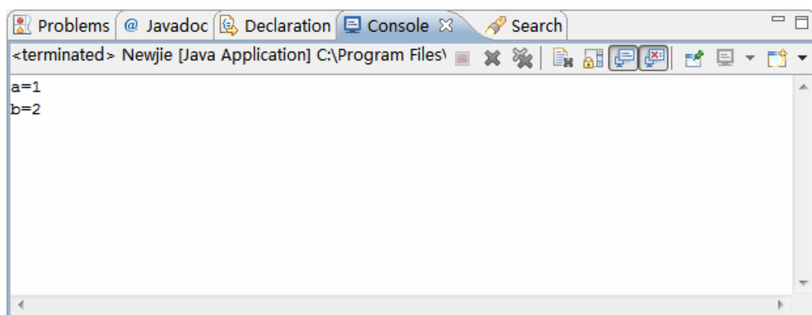
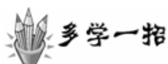


图 9-14　定义接口



初学者可以将接口理解成抽象类，抽象类如何定义方法，接口就可以如何定义方法。在上面的代码中，用户需要注意，“implements”是用于实现接口的关键字，相当于类继承父类的关键字，在后面将会详细讲解。

9.3.2　接口里的量和方法

在接口内定义变量，只能使用 public、static 和 final 关键字，因此在接口中只能声明常量，不能声明变量。在接口里，有的方法是必须抽象方法，在本节中将详细讲解。

（1）接口里的量

在接口中只能有常量，其主要原因是这样能保证实现该接口的所有类可以访问相同的常量。下面通过一个实例进行讲解。

实例 46：将定义一个接口在接口里面编写常量

下面将定义一个接口，在接口里面编写常量，其代码见“光盘：源代码/第 9 章/Jiechang.java”：

```
public interface Jiechang
{
 int a=100;
 int b=200;
 int c=323;
 int d=234;
 int f=523;
 void print();
 void print1();
}

class Jiedo implements Jiechang
{
 public void print()
 {
     System.out.println(a+b);
 }
```

```
     public void print1()
     {
         System.out.println(c+d+f);
     }
    }

    class Jie
    {
     public static void main(String args[])
     {
         Jiedo a1=new Jiedo();
         a1.print();
         a1.print1();
     }
    }
```

运行代码，得到如图 9-15 所示的结果。

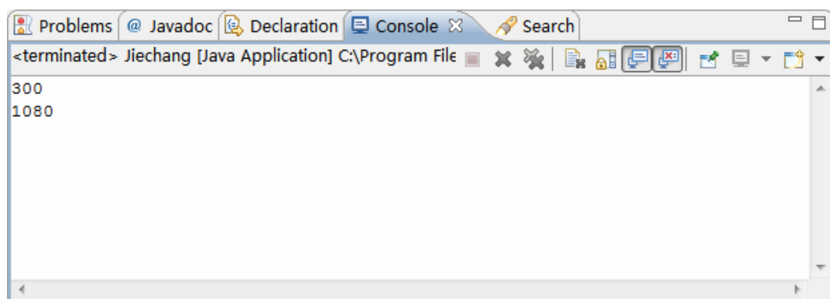


图 9-15　接口里的量

多学一招

在上面的实例中讲解了在接口中只能定义常量，许多人肯定不相信，下面将通过修改上面的代码进行讲解，让读者明白，接口里只能有常量，不能有变量，其代码见“光盘：源代码/第 9 章/ jiecuo.java”：

```
    interface Example
    {
     int a=12;
     int b=22;
     int c=32;

     void print(int x);
    }
    class Example1 implements Example
    {
     public void print(int x)
```

```
        {
            if(x>a)
            {
                b=x;
                c=x+1;
                a=b+c;
                System.out.println(b);
                System.out.println(c);
            }
        }
    }
    class jiecuo
    {
     public static void main(String args[])
     {
         Example1 a1=new Example1();
         a1.print(4);
     }
    }
```

这个程序是不能执行的，在编译时就会出现错误，如图 9-16 所示。

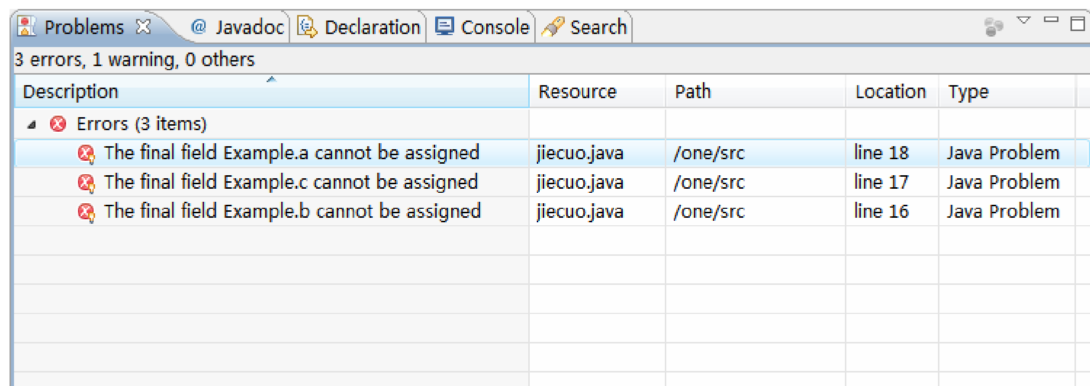


图 9-16　编译错误

（2）接口里的方法

接口里的方法都是抽象的或者公有的，在声明方法的时候，可以省掉 public、abstract 关键字，因为它的方法都是公有和抽象的，不需要关键字修饰，当然添加修饰符也没有错。

代码 43：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 9 章/ cuofang.java”：

```
    interface newjie
    {
     void print();
     public void print1();
     abstract void print2();
     public abstract void print3();
```

```
     abstract public void print4();
    }

    class newjie1 implements newjie
    {
     public void print()
     {
         System.out.println("newjie 接口里第一个方法没有修饰符");
     }

     public void print1()
     {
         System.out.println("newjie 接口里第二个方法有修饰符 public");
     }

     public  void print2()
     {
         System.out.println("newjie 接口里第三个方法有修饰符 abstract");
     }

     public  void print3()
     {
        System.out.println("newjie 接口里第四个方法有修饰符 public 和 abstract");
     }

     public void print4()
     {
        System.out.println("newjie 接口里第五个方法有修饰符 abstract 和 public");
     }
    }

    class coufang
    {
     public static void main(String args[])
     {
         newjie1 a1=new newjie1();
         a1.print();
         a1.print1();
         a1.print2();
         a1.print3();
         a1.print4();
     }
    }
```

运行代码，得到如图 9-17 所示的结果。

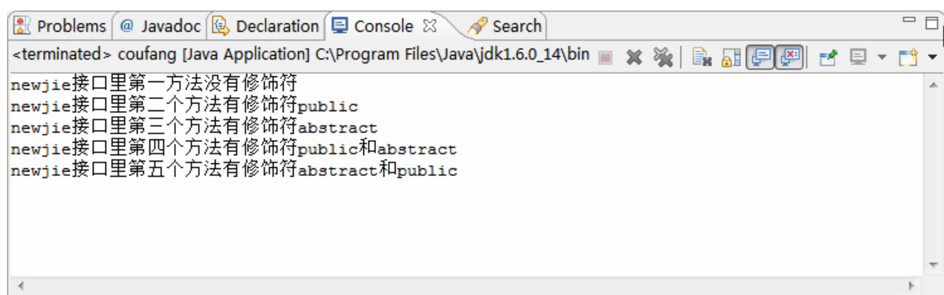


图 9-17　接口里的方法

提示：在上面的程序中，定义了接口并在接口里定义了五个方法，这五个方法实际上是相同的，在编写程序时，用户大可只用第一种方式编写。

9.3.3　接口的实现

实际上在前面的学习中，用户已经理解了接口的实现，在接口实现的过程中，要保证能为所有的接口提供实现的功能，能遵循重写的所有规则，能保持相同的返回的数据类型。实现接口的方法如下：

```
[<修饰符>] class<类名> implements <接口名>
{
……
}
```

实例 47：编写一个类去实现一个接口

编写一个类，去实现一个接口，让这个类具备这个接口的一些功能，其代码见“光盘：源代码/第 9 章/ jieshi.java”：

```
interface JieOne
{
 int add(int a,int b);
}

interface JieTwo
{
 int sub(int a,int b);
}

interface JieThree
{
 int mul(int a,int b);
}

interface JieFour
{
 int umul(int a,int b);
```

```
    }

    class JieDuo implements JieOne,JieTwo,JieThree,JieFour
    {
     public int add(int a,int b)
     {
         return a+b;
     }
         public int sub(int a,int b)
     {
         return a-b;
     }
         public int mul(int a,int b)
     {
         return a*b;
     }
     public int umul(int a,int b)
     {
         return a/b;
     }
    }

    class jieshi
    {
     public static void main(String args[])
     {
         JieDuo aa=new JieDuo();
         System.out.println("a+b="+aa.add(2400,1200));  //提供具体实现方法
         System.out.println("a-b="+aa.sub(2400,1200));  //提供具体实现方法
         System.out.println("a*b="+aa.mul(2400,1200));  //提供具体实现方法
         System.out.println("a/b="+aa.umul(2400,1200)); //提供具体实现方法
     }
    }
```

运行代码，得到如图 9-18 所示的结果。

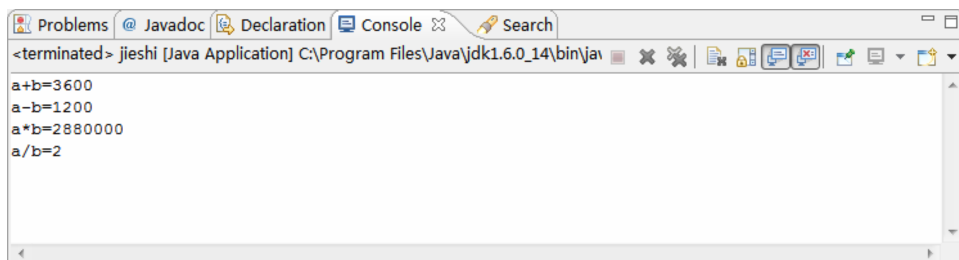
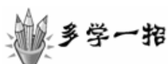


图 9-18　实现接口



定义一个类，去实现多个接口。实现多个接口十分简单，只需要在接口与接口之间用逗号隔开即可。接口和类十分相似，类可以继承，接口也可以继承。下面通过一段代码讲解接口的继承，其代码见“光盘：源代码/第 9 章/ jieji.java”：

```
interface Aone{
 void add(int a,int b);
}
interface Atwo{
 void sub(int a,int b);
}
//创建 Athree 接口继承了 Aone，Atwo 接口
interface Athree extends Aone,Atwo{
 void mul(int a,int b);
}
class JieXian implements Athree
{
 public void add(int a,int b)
 {
     System.out.println(a+b);
 }
 public void sub(int a,int b)
 {
     System.out.println(a-b);
 }
     public void mul(int a,int b)
 {
     System.out.println(a*b);
 }
}
class jieji
{
 public static void main(String args[])
 {
     JieXian aa=new JieXian();
     aa.add(122,232);
     aa.sub(122,232);
     aa.mul(122,232);
 }
}
```

运行代码，得到如图 9-19 所示的结果。

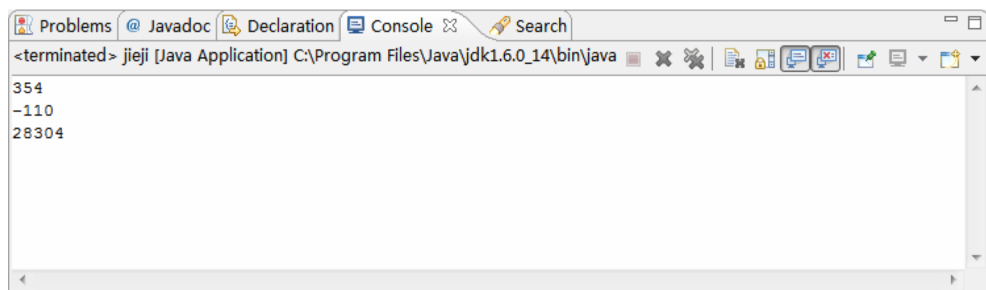


图 9-19　接口的继承

9.3.4　接口的引用

在编写程序时，用户可以建立接口类型的引用变量，接口的引用变量能够存储一个指向对象的引用值，这个对象可以实现任何该接口的类的实例，用户可以通过接口调用该对象的方法，这些方法必须是类中的抽象方法。

代码 44：下面通过一段代码进行讲解，其代码见“光盘：源代码/第 9 章/ jieyin.java”：

```
interface diyijie
{
 int add(int a,int b);
}
interface dierjie
{
 int sub(int a,int b);
}
interface disanjie
{
 int mul(int a,int b);
}
interface disijie
{
 int umul(int a,int b);
}
class jiekouniu implements diyijie,dierjie,disanjie,disijie
{
 public int add(int a,int b)
 {
     return a+b;
 }
 public int sub(int a,int b)
 {
     return a-b;
 }
     public int mul(int a,int b)
 {
```

```
            return a*b;
        }
            public int umul(int a,int b)
        {
            return a/b;
        }
    }
    class jieyin
    {
     public static void main(String args[])
     {
         jiekouniu aa=new jiekouniu();
         //接口的引用执行对象的引用
         diyijie  bb=aa;
         dierjie  cc=aa;
         disanjie  dd=aa;
         disijie ee=aa;
         //对象引用并调用方法
         System.out.println("a+b="+aa.add(14,22));
         System.out.println("a-b="+aa.sub(42,32));
         System.out.println("a*b="+aa.mul(44,22));
         System.out.println("a/b="+aa.umul(24,22));
         System.out.println("a+b="+bb.add(23,42));
         System.out.println("a-b="+cc.sub(32,12));
         System.out.println("a*b="+dd.mul(42,24));
         System.out.println("a/b="+ee.umul(342,22));
             }
    }
```

运行代码，得到如图 9-20 所示的结果。

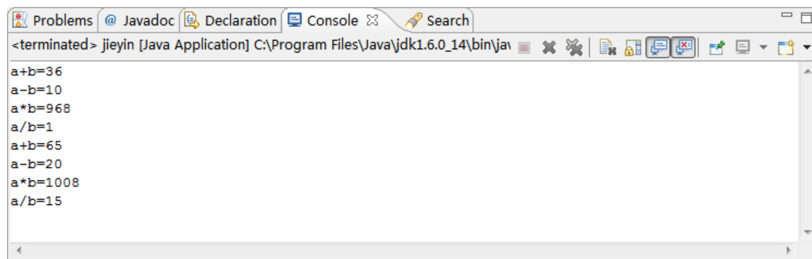


图 9-20　接口的引用

9.4　疑难问题解析

本章详细介绍了类的继承、重写、重载的基本知识。本节将对本章中较难理解的问题进

行讲解。

读者疑问：接口和类的许多特性十分相似，在开发 Java 程序时为什么还要使用接口？只用类不能实现所有功能吗？

解答：类是 Java 程序设计的核心，它虽然功能强大，但并不能概括所有的特性。接口是特殊的类，它和抽象类特性几乎相同，它的诞生主要是为了弥补 Java 程序类的继承特性，因为 Java 程序的类是单一继承，但是一个类却可以实现多个接口。

读者疑问：本章讲解了重写和重载，它们之间有什么区别呢？

解答：重写和重载虽然只有一个字的差别，但在功能上却有着天壤之别。重写一般应用在继承的子类中，而重载只会出现在一个类中。重写是覆盖分类的方法，而重载是同一类中有多个方法名相同，功能相近，但参数不同的方法。初学者只需通过一个简单的口诀即可理解重写重载，"继承可重写，方法可重载。"



职场点拨——模块化设计的重要性

模块是指程序中的一段代码，该段代码能实现程序中的某一功能并能独立或半独立运行。在大型程序编写中，模块话的运用是不可避免的。从面向对象编程思想被推出后，模块设计思想就成为了一个主流的软件开发模式。

在设计大型程序时，常常要将整个问题分解为若干个小问题，必要时还要将小问题再次分解为更小的若干问题，每个小问题编写成独立的源文件，最后将所有的源文件连接起来组合成一个大程序。也就是说，一个程序往往由多个源文件组成，那么构成一个程序的各个相对独立的源文件通常称为模块。这样把一个程序分成多个功能相对独立的程序模块分别编制、调试后，再用连接程序把它们连接在一起生成一个完整的程序设计的方法称为模块化程序设计。

模块的划分应该是灵活的，但不应是程序的等分处理，应使各模块具有相对的独立性和完整性，可以单独编程、调试，但也要考虑各个模块之间的联系。模块划分是一个自上而下的过程。主模块是一个总控模块，首先确定主要的模块，也就是说，要把总任务划分成几个主要的子任务。一般来说，可以分成输入任务、输出任务和一个或多个进行处理或计算的子任务。在划分子模块的过程中应该明确每个模块的功能、数据结构及相互之间的关系。第二步，对这些主要的子模块根据需要再划分成下一层的子模块。第三步，重复上述过程，一直到程序分成易于理解和易于实现的小模块为止。

1．模块化设计的优点

模块化程序设计的优点如下:

1）速度快。

2）可维护性与可读性强。

3）可移植性强。

2．模块划分的方法

当前模块划分的方法有 2 种，具体说明如下:

（1）层次图

层次图是表示模块与模块之间关系的方块图。层次图的顶端是主模块，即一个总控制块，直接控制位于其下一层的各个模块的执行，而各主要的子模块再去控制其下一层的子模块。

（2）模块说明

模块说明是对模块的功能、算法、模块输入和输出以及它们的数据结构的简单说明。应该考虑程序中哪些数据放在公共数据区，供所有模块访问，哪些数据可在有直接从属关系的模块间传送。

3．模块化的编程中的应用

我们以一个例子来说明，例如你打开一个典型的 Web 站点，会发现整个站点是由不同功能的模块构成的。一个典型项目程序的基本结构如下图所示：



再打个比方，有一个最基本的会员登录系统，可以由以下模块构成：

1）表单模块：显示用户的登录表单。

2）登录验证模块：验证用户输入的信息是否合法。

当以后系统升级时，例如验证选项的变化，就只需对某一处进行修改即可。这样便减轻了后期维护负担，提高了工作效率。

第 10 章 异 常 处 理

在程序中出现异常是在所难免的，异常是什么呢？所谓异常，就是程序在运行时发生的错误或者不正常的情况。异常对程序员来说是一件很麻烦的事情，需要对整个程序进行检测以处理异常。不过 Java 有一个优点，它可以自动检测异常，对异常进行捕获，并且通过程序对异常进行处理。本章将详细讲解如何处理异常。本章主要内容如下：

- 什么是异常。
- 异常处理方式。
- 异常抛出。
- 自定义异常。
- 职场点拨——分析老板们的不同特点。

2010 年 X 月 XX 日，暴雨

今天我很郁闷，项目总监做出新项目的策划，让项目经理和 HR 经理合作完成。项目经理老李马上要退休了，几乎已经不管事情了，直接就把任务甩给了我。我只是一个程序员，对公司战略和项目规划没有任何经验，让我从何下手啊！


一问一答

小菜：“真烦人啊，这样的领导，什么也不管……”

Wisdom：“呵呵，你对老李的抱怨是很正常的，但是你也要体谅他，即将退休的他虽然看似懒散不求上进，但反过来也是在给你机会。具体怎样体谅呢？就需要你认真分析这些老板们的特点了。”

小菜：“也是，像我这样的‘完人’已经很稀缺了，我也不能强求他什么了。”

Wisdom：“不，职场中人无完人，任何人都有优点和缺点，包括你我在内。我们需要做的是发挥自己的优点，尽量避免自己的缺点。同样，在 Java 程序中，错误也是不可避免的，不管多么优秀的程序员，都有可能出错，为了很好地解决这个问题，Java 推出了异常处理机制。”

10.1 什么是异常

在程序设计中，异常处理就是提前编写程序处理可能发生的意外，如聊天工具需要连接

网络，首先要做的就是检查网络，对网络的各个程序进行捕获，然后针对各种情况编写不同的程序。本节将对异常进行系统地讲解。

10.1.1　认识异常

在编程过程中，应当尽可能避免错误和异常发生，对于不可避免、不可预测的情况，则应思考应对方法。Java 中的异常可以用对象来表示。Java 对异常的处理是按异常分类处理的，异常的分类很多，每种异常都对应一个类型，每个异常也都对应一个异常（类的）对象。

异常的对象是从哪里来的呢？

异常主要有两个来源，一是 Java 运行时环境自动抛出系统生成的异常，不管你是否愿意捕获和处理，它总要被抛出，比如除数为 0 的异常。二是程序员自己抛出的异常，这个异常可以是程序员自己定义的，也可以是 Java 语言中定义的，如使用 throw 关键字抛出异常，这种异常常用来向调用者汇报异常的一些信息。

异常是针对方法来说的，抛出、声明抛出、捕获和处理异常都是在方法中进行的。

Java 异常处理通过 try、catch、throw、throws、finally 五个关键字进行管理。基本过程是用 try 语句块包住要监视的语句，如果在 try 语句块内出现异常，则异常会被抛出，catch 语句块会捕获到这个异常并做处理。还有以部分系统生成的异常在 Java 运行时自动抛出，也可以通过 throws 关键字在方法上声明该方法要抛出异常，然后在方法内部通过 throw 抛出异常对象。finally 语句块会在方法执行 return 之前执行，一般结构如下：

```
try
{
程序代码
}catch(异常类型 1 异常的变量名 1)
{
程序代码
}catch(异常类型 2 异常的变量名 2)
{
程序代码
}finally
{
程序代码
}
```

10.1.2　Java 提供的异常处理类

在 Java 的程序设计中有一个 lang 包，包中有一个专门处理异常的类 Throwable 类，它是所有异常的父类，每一个异常的类都是它的子类。Error 和 Exception 这两个类也十分重要，用得也较多，前者是用来定义那些通常情况下不希望被捕获的异常，后者是程序能够捕获的异常情况，如表 10-1 所示。

表 10-1　算术运算符

异常类名称	异常类含义
ArithmeticException	算术异常类
ArratIndexOutOfBoundsExeption	数组小标越界异常类
ArrayStroeException	将与数组类型不兼容的值赋值给数组元素时抛出的异常
ClassCastException	类型强制转换异常类
ClassNotFoundException	为找到相应大类异常
EOFEException	文件已结束异常类
FileNotFoundException	文件未找到异常类
IllegalAccessException	访问某类被拒绝时抛出的异常类
InstantiationException	试图通过newInstance（）方法创建一个抽象类或抽象接口的实例时抛出异常类
异常类名称	异常类含义
IOEException	输入输出抛出异常类
NegativeArraySizeException	建立元素个数为负数的异常类
NullPointerException	空指针异常
NumberFormatException	字符串转换为数字异常类
NoSuchFieldException	字段未找到异常类
NoSuchMethodException	方法未找到异常类
SecurityException	小应用程序执行浏览器的安全设置禁止动作时抛出的异常类
SQLException	操作数据库异常类
StringIndexOutOfBoundsException	字符串索引超出范围异常类

10.2　异常处理方式

异常处理是为了防止产生未知错误所采取的处理措施，Java 程序为开发人员提供了一套完美的处理异常的方法，下面将详细讲解 Java 程序处理异常的方法。

10.2.1　使用 try…catch 处理异常

在编写 Java 程序时，需要处理的异常一般是放在 try 代码块里，然后创建 catch 代码块进行捕获和处理。下面通过一个实例进行讲解。

实例 48：使用 try…catch 进行捕获并处理

编写一个类，使用 try…catch 进行捕获并对它进行处理，其代码见“光盘：源代码/第 10 章/YiOne1.java”：

```
public class YiOne1
 {
 public static void main(String args[])
 {

     int x,y;
     try
```

```
        {
            x=0;
            y=5/x;
            System.out.println("需要检验的程序");
        }
        catch(ArithmeticException e)
        {
            System.out.println("发生了异常，分母不能为零");
        }
        System.out.println("程序运行结束");
    }
  }
```

上面这段代码中有着明显的错误，就是算术式子里有一个分母为零，就连小学生都知道分母是不能为零的。这段代码需要放在 try 代码块里，然后通过 catch 里的代码对它进行处理。运行代码，得到如图 10-1 所示的结果。

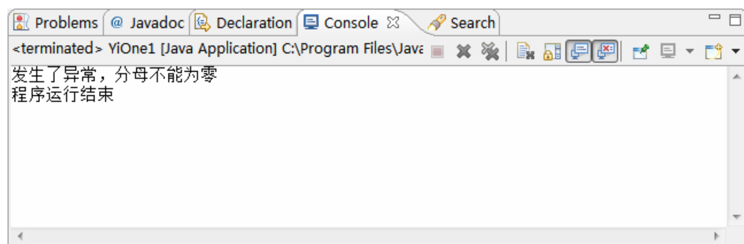


图 10-1　处理异常

多学一招

上面这段代码是用户自己编写对它进行处理的，实际上可以将之交由系统处理。修改上面的代码，得到如下的代码:

```
  public class YiOne1
  {
   public static void main(String args[])
   {
        int x,y;
        try
        {
              //捕获异常
              x=0;
              y=5/x;
              System.out.println("需要检验的程序");
        }
        catch(ArithmeticException e)
        {
```

```
            //处理异常
            System.out.println(e);
        }
        System.out.println("程序运行结束");
    }
```

运行代码，得到如图 10-2 所示的结果。

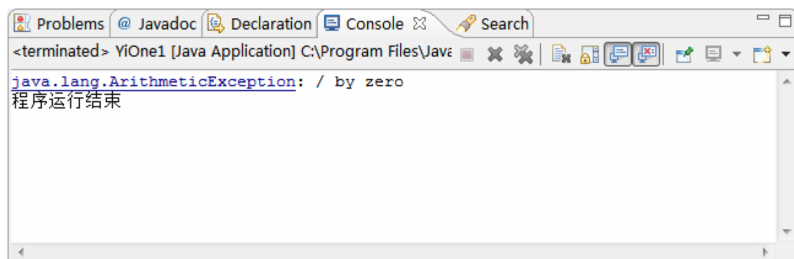


图 10-2　系统处理异常

10.2.2　处理多个异常

对于同一件事情不能正常处理，可能是因为出现的异常不止一种。下面通过一段代码讲解如何处理多个异常，其代码见“光盘：源代码/第 10 章/Yitwo1.java”：

```
public class Yitwo1
 {
 public static void main(String args[])
 {
            int [] a=new int[5];
            try
            {
                    a[6]=123;
                    System.out.println("需要检验的程序");
            }
            catch(ArrayIndexOutOfBoundsException e)
            {
                    System.out.println("发生 ArrayIndexOutOfBoundsException
了异常");
            }
            catch(ArithmeticException e)
            {
                    System.out.println("发生了 ArithmeticException 异常");
            }
            catch(Exception e)
            {
                    System.out.println("发生了 Exception 异常");
            }
```

```
                System.out.println("结束");
        }
    }
```

运行代码，提示发现异常，如图 10-3 所示。

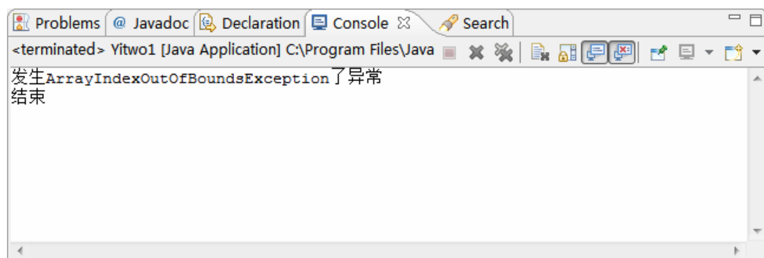


图 10-3　处理多个异常

10.2.3　在异常中使用 finally 关键字

使用 try…catch 时，配以关键字 finally 可以增强处理异常的功能。不管程序有无异常发生，都将执行 finally 语句块的内容，这样使得一些不管在任何情况下都必须执行的步骤被执行，可保证程序的健壮性。

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 10 章/Yitwo2.java”：

```
    public class Yitwo2
    {
     public static void main(String args[])
     {
                        try
                {
                    int age=Integer.parseInt("25L");//抛出异常
                    System.out.println("输出 1");
                }
                catch(NumberFormatException e)
                {
                    int b=8/0;
                    System.out.println("请输入整数年龄");
                    System.out.println("错误"+e.getMessage());
                }
                finally {
                    System.out.println("输出 2");
                          }
                System.out.println("输出 3");
            }
    }
```

运行代码，得到如图 10-4 所示的结果。

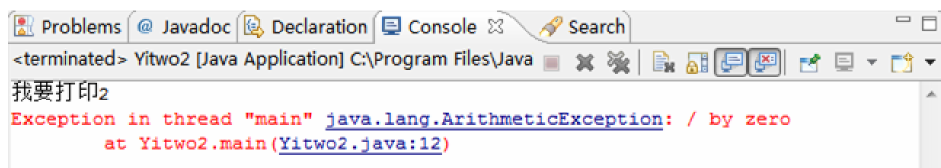


图 10-4　使用 finally 关键字

10.3　将异常抛出

在很多时候，程序不会立即处理异常，而是将它抛出，交给父类，由父类对它进行处理。这一做法在编程的过程中经常会用到。

10.3.1　使用 throws 将异常抛出

声明异常是指一个方法不处理异常，而是调用层次向上传递，谁调用这个方法，这个异常就由谁处理。throws 是其中的一个，它的声明格式如下：

```
Void methodName（int a）throws Exception
{
}
```

实例 49：处理异常

编写一个程序使用 throws 关键字将异常抛出，其代码见“光盘：源代码/第 10 章/YiThree1.java”：

```
public class YiThree1
 {
   public void methodName(int x) throws
    ArrayIndexOutOfBoundsException,ArithmeticException
      {
           System.out.println(x);
           if(x==0)
           {
               System.out.println("没有异常");
               return;
           }
           else if(x==1)
           {
               int [] a=new int[3];
               a[3]=5;
           }
           else if(x==2)
           {
               int i=0;
```

```
                int j=5/i;
            }
        }
        public static void main(String args[])
        {
            YiThree1 ab=new YiThree1();
            try
            {
                ab.methodName(0);
            }
            catch(Exception e)
            {
                System.out.println("异常:"+e);
            }
            try
            {
                ab.methodName(1);
            }
            catch(ArrayIndexOutOfBoundsException e)
            {
                System.out.println("异常:"+e);
            }
            try
            {
                ab.methodName(2);
            }
            catch(ArithmeticException e)
            {
                System.out.println("异常:"+e);
            }
        }
    }
```

运行代码，得到如图 10-5 所示的结果。

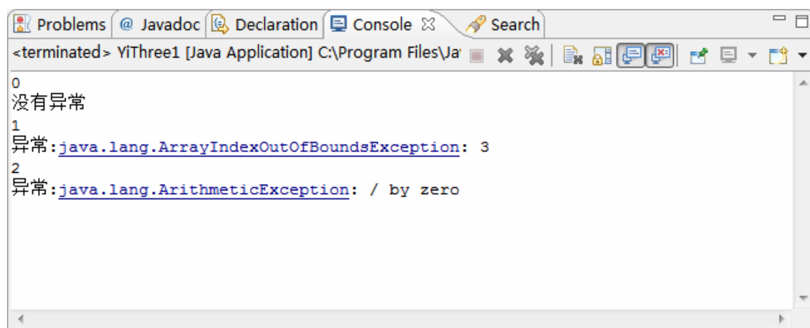
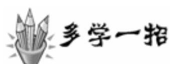


图 10-5　使用 throws 抛出异常



在上面的程序中使用 throws 关键字抛出了异常。下面再讲解两段代码，以扩宽读者对异常处理更深层次的认识。

第一段代码见“光盘：源代码/第 10 章/Yifour1.java”：

```
public class yifour1 extends Exception
//自定义异常，这里不要求掌握，在后面一节会讲解。
{
  private String name;
  public  yifour1(String name)
{
      this.name=name;
  }
  public String getMessage()
{
      return this.name;
  }
}
```

第二段代码见“光盘：源代码/第 10 章/Yifour2.java”：

```
public class yifour2
 {
   public static int check(String strage) throws yifour1
   {
           int age=Integer.parseInt(strage);
           if(age<0)
           throw new yifour1("年龄不能为负数！");
           return age;
       }
       public static void main(String[] args)
   {
           try{
               int myage=check("-101");
               System.out.println(myage);
           }catch(NumberFormatException e){
               System.out.println("数据格式错误！");
               System.out.println("原因："+e.getMessage());
           }catch(yifour1 e){
               System.out.println("数据逻辑错误！");
               System.out.println("原因："+e.getMessage());
           }
      }
}
```

运行代码，得到如图 10-6 所示结果。

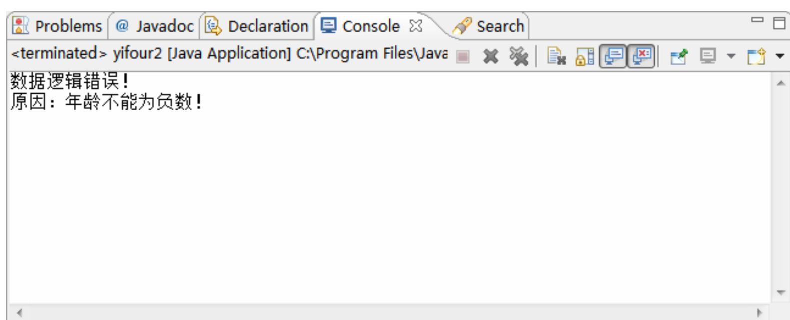


图 10-6　使用 throws 抛出异常

10.3.2　使用 throw 将异常抛出

抛出异常时可以使用 throw 关键字，把它抛给上一级调用的异常。抛出的异常可以是异常引用，也可以是异常对象。

实例 50：使用 throw 关键字将异常抛出

编写一个程序使用 throw 关键字将异常抛出，其代码见“光盘：源代码/第 10 章/YiFour.java”：

```
public class YiFour
{
   public static void main(String args[])
   {
       try
       {
           throw new ArrayIndexOutOfBoundsException();
       }
       catch(ArrayIndexOutOfBoundsException aoe){
           System.out.println("异常:"+aoe);
       }
       try
       {
           throw new ArithmeticException();
       }
      catch(ArithmeticException ae)
      {
           System.out.println("异常:"+ae);
      }
}
}
```

运行代码，得到如图 10-7 所示的效果。

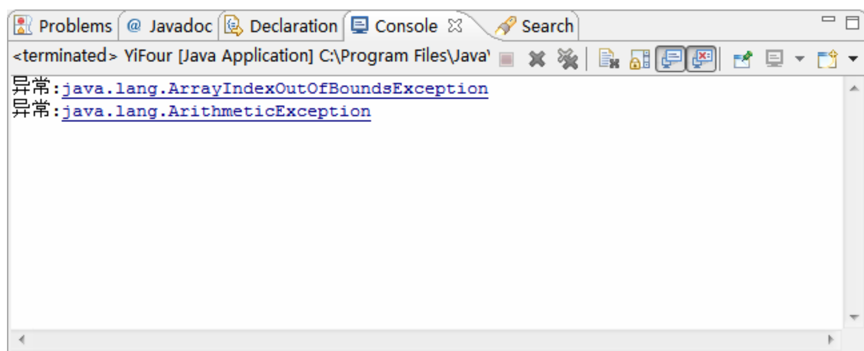


图 10-7　执行结果

多学一招

在上面的程序中使用 throw 关键字抛出了异常，下面再讲解一段代码，以扩宽读者对异常处理更深层次的认识。其代码见“光盘：源代码/第 10 章/YiF1.java”：

```
public class YiF1
 {
   void methodName()
   {
        try
        {
            throw new ArrayIndexOutOfBoundsException();
        }
        catch(ArrayIndexOutOfBoundsException aoe)
        {
            throw aoe;
        }
   }
   public static void main(String args[])
   {
        YiF1 yc=new YiF1();
        try
        {
            yc.methodName();
        }
        catch(ArrayIndexOutOfBoundsException aoe)
        {
            System.out.println("异常:"+aoe);
        }
   }
}
```

运行代码，得到如图 10-8 所示的结果。

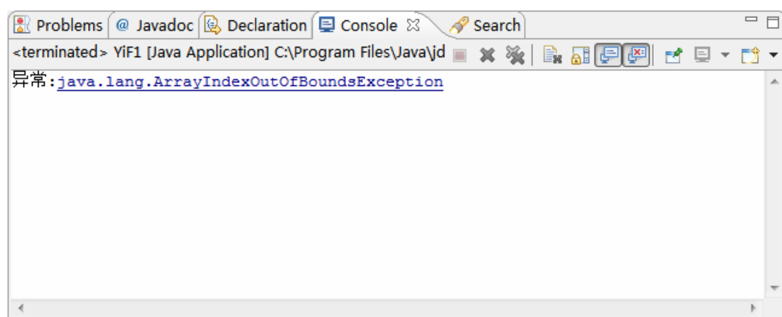


图 10-8　出现的异常

10.4　自定义异常

前面讲解的异常都是系统自带、自己处理的异常。但是很多时候需要程序员自定义异常，在遇到这样的异常时再进行处理。创建自定义异常只需要继承 Throwable 类或者它的子类 Exception 即可。自定义异常让系统把它看成一种异常来对待，由于自定义异常继承 Throwable 类，所以也继承了它的方法，它的方法如下：

- fillInStack Trace()。
- getCause()。
- getLocalizedMessage()。
- getMessage()。
- getStack Trace()。
- initCause(Throwable cause)。
- printStack Trace()。
- printStack Trace(PrintStream s)。
- printStack Trace(PrintWriter s)。
- setStack Trace(Stack TraceElement[] stack Trace)。
- toString()。

实例 51：自定义异常并将其解决

下面编写几段程序，其中自定义了异常并将其解决，其代码分别罗列。

第一段代码见“光盘：源代码/第 10 章/YiZone1.java”：

```
public class YiZone1 extends Exception
{
public YiZone1()
{
super();
}
public YiZone1(String msg)
{
super(msg);
```

```
}
public YiZone1(String msg, Throwable cause)
 {
super(msg, cause);
}
public YiZone1(Throwable cause)
{
super(cause);
}
}
```

第二段代码见“光盘：源代码/第 10 章/MyYi.java”：

```
public class MyYi extends Throwable
{
public MyYi()
{
super();
}
public MyYi(String msg)
{
super(msg);
}
public MyYi(String msg, Throwable cause)
{
super(msg, cause);
}
public MyYi(Throwable cause)
{
super(cause);
}
}
```

第三段代码见“光盘：源代码/第 10 章/MyyiT.java”：

```
public class MyyiT
{
public static void firstException() throws MyYi{
throw new MyYi("\"firstException()\" method occurs an exception!");
}

public static void secondException() throws MyYi{
throw new MyYi("\"secondException()\" method occurs an exception!");
}
public static void main(String[] args)
{
try {
   MyyiT.firstException();
```

```
        MyyiT.secondException();
    } catch (MyYi e2){
    System.out.println("Exception: " + e2.getMessage());
    e2.printStackTrace();
    }
    }
    }
```

运行代码，得到如图 10-9 所示的结果。

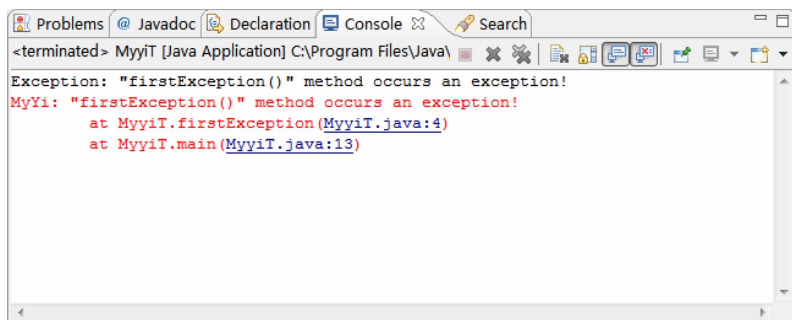
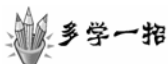


图 10-9　执行结果



在上面的程序中，有了自定义的异常，下面再通过一段代码进行讲解，以加深读者对自定义异常的理解，其代码见“光盘：源代码/第 10 章/YiTai.java”：

```
class MyException extends Exception
{
   private int chengji;
   MyException(int a)
   {
         chengji=a;
   }
   public String toString()
   {
         return "异常:"+chengji;
   }
}
public class YiTai
{
   void methodName(int a) throws MyException
   {
         System.out.println(a);
         if(a<60)
         {
               throw new MyException(a);
```

```
            }
            else
            {
                System.out.println("正常");
            }
        }
        public static void main(String args[])
        {
            YiTai yc=new YiTai();
            try
            {
                yc.methodName(75);
                yc.methodName(55);
            }
            catch(MyException e)
            {
                System.out.println(e);
            }
        }
    }
```

运行代码，得到如图 10-10 所示的结果。

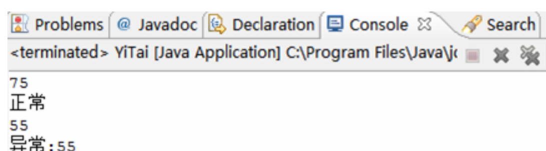


图 10-10　执行结果

10.5　异常处理的陋习

作为 Java 程序员，一定要养成异常处理的好习惯，不要养成异常处理陋习。本小节将通过一段代码进行讲解，让读者彻底摆脱陋习，将 Java 代码书写更规范，更具有健壮性。其代码如下：

```
OutputStreamWriter out=...
java.sql.Connection conn=...
try { // ⑸
  Statement stat=conn.createStatement();
  ResultSet rs=stat.executeQuery(
    "select uid, name from user");
  while (rs.next())
  {
    out.println("ID: " + rs.getString("uid") // ⑹
```

```
10      "，姓名：" + rs.getString("name"));
11   }
12   conn.close(); // ⑶
13   out.close();
14 }
15 catch(Exception ex) // ⑵
16 {
17   ex.printStackTrace(); //⑴，⑷
18 }
```

如果读者不能找出代码中存在的全部 6 个问题，则说明在程序的书写上还存有陋习，下面进行详细讲解。

10.5.1　丢弃异常

在上面的代码中的第 15 行～18 行，这段代码捕获了异常却不做任何处理，可以算得上 Java 编程中的杀手。从问题出现的频繁程度和危害程度来看，它也许可以和 C/C++程序的一个恶名远播的问题—不检查缓冲区是否已满相提并论。如果发现这种丢弃（不是抛出）异常的情况，可以百分之九十九地肯定代码存在问题（只有在极少数情况下，这段代码有存在的理由，但最好加上完整的注释，以免引起误解）。

这段代码的错误在于异常几乎总是意味着某些事情不对劲，或者说至少发生了某些不寻常的事情，作为程序员，不应该对程序发出的求救信号保持沉默和无动于衷。只是调用一下 printStackTrace 算不上“处理异常”。虽然调用 printStackTrace 对调试程序有帮助，但程序调试阶段结束之后，printStackTrace 就不应再在异常处理模块中担负主要责任了。

丢弃异常的情形非常普遍。打开 JDK 的 ThreadDeath 类的文档，可以看到下面这段说明，“特别地，虽然出现 ThreadDeath 是一种‘正常的情形’，但 ThreadDeath 类是 Error 而不是 Exception 的子类，因为许多应用会捕获所有的 Exception，然后丢弃它不再理睬。”，这段话的意思是，虽然 ThreadDeath 代表的是一种普通的问题，但鉴于许多应用会试图捕获所有异常，然后不予以适当处理，所以 JDK 把 ThreadDeath 定义成了 Error 的子类，因为 Error 类代表的是一般的应用不应该去捕获的严重问题。可见，丢弃异常这一坏习惯是如此常见，它甚至已经影响到了 Java 本身的设计。

那么应该怎样改正呢？主要有 4 个选择，其介绍如下：

- **处理异常**：针对该异常采取一些行动，例如修正问题、提醒某个人或进行其他一些处理，要根据具体的情形确定应该采取的动作。再次说明，调用 printStackTrace 算不上已经“处理好了异常”。
- **重新抛出异常**：处理异常的代码在分析异常之后，认为自己不能处理它，重新抛出异常也不失为一种选择。
- **把该异常转换成另一种异常**：大多数情况下，这是指把一个低级的异常转换成应用级的异常（其含义更容易被用户了解的异常）。
- **不要捕获异常**。

提示：既然捕获了异常，就要对它进行适当处理。不要捕获异常之后又把它丢弃，不予理睬，这种处理异常的方法将会直接影响 Java 程序的健壮性。

10.5.2 不指定具体的异常

代码第 15 行虽然交代了异常，但是指定并不具体。许多时候人们会被这样一种“美妙的”想法吸，即用一个 catch 语句捕获所有的异常。最常见的情形就是使用 catch(Exception ex)语句。但实际上，在绝大多数情况下，这种做法不值得提倡。这是为什么呢？

要理解其原因，必须回顾一下 catch 语句的用途。catch 语句表示预期会出现某种异常，而且希望能够处理该异常。异常类的作用就是告诉 Java 编译器想要处理的是哪一种异常。由于绝大多数异常都直接或间接从 java.lang.Exception 派生，catch(Exception ex)就相当于说想要处理几乎所有的异常，这是很不实际的一种做法。

再来看看前面的代码，真正想要捕获的异常是什么呢？最明显的一个是 SQLException，这是 JDBC 操作中常见的异常，另一个可能的异常是 IOException，因为它要操作 OutputStreamWriter。显然，在同一个 catch 语句块中处理这两种截然不同的异常是不合适的。如果用两个 catch 语句块分别捕获 SQLException 和 IOException 就要好多了。这就是说，catch 语句应当尽量指定具体的异常类型，而不应该指定涵盖范围太广的 Exception 类。

另一方面，除了这两个特定的异常，还有其他许多异常也可能出现。如果由于某种原因，executeQuery 返回了 null，该怎么办？答案是让它们继续抛出，即不必捕获也不必处理。实际上，在编程过程中不能也不应该去捕获可能出现的所有异常，程序的其他地方还有捕获异常的机会，直至最后由 JVM 处理。

提示：在 catch 语句中尽可能指定具体的异常类型，必要时使用多个 catch。不要试图处理所有可能出现的异常。

10.5.3 占用资源不释放

读者仔细阅读代码的第 3 行～14 行，就有占用资源不释放的情况。

异常改变了程序正常的执行流程，这个道理虽然简单，却常常被人们忽视。如果程序用到了文件、Socket、JDBC 连接之类的资源，即使遇到了异常，也要正确释放占用的资源。为此，Java 提供了一个简化这类操作的关键词 finally。

finally 是样好东西，不管是否出现了异常，finally 保证在 try/catch/finally 代码块结束之前，执行清理任务的代码总是有机会执行。遗憾的是有些人却不习惯使用 finally。

当然，编写 finally 代码块应当多加小心，特别是要注意在 finally 代码块之内抛出的异常，这是执行清理任务的最后机会，尽量不要再有难以处理的错误。

提示：保证所有资源都被正确释放。充分运用 finally 关键词。

10.5.4 不说明异常的详细信息

仔细观察代码第 3 行～18 行，如果循环内部出现了异常，会发生什么事情？是否可以得到足够的信息判断循环内部出错的原因？答案是不能，至多只能知道当前正在处理的类究竟发生了某种错误，但却不能获得任何信息判断导致当前错误的原因。

printStackTrace 的堆栈跟踪功能显示出程序运行到当前类的执行流程，但只提供了一些最基本的信息，未能说明实际导致错误的原因，同时也不易解读。

因此，在出现异常时，最好能够提供一些文字信息，例如当前正在执行的类、方法和其他状态信息，包括以一种更适合阅读的方式整理和组织 printStackTrace 提供的信息。

提示：在异常处理模块中提供适量的错误原因信息，组织错误信息使其易于理解和阅读。

10.5.5　过于庞大的 try 块

仔细观察代码第 3 行～14 行，经常可以看到有人把大量的代码放入单个 try 块，实际上这不是好习惯。这种现象之所以常见，原因就在于有些人图省事，不愿花时间分析一大块代码中哪几行代码会抛出异常、异常的具体类型是什么。把大量的语句装入单个巨大的 try 代码块，类似出门旅游时把所有日常用品塞入一个大箱子，虽然东西是带上了，但要找出来可不容易。一些新手常常把大量的代码放入单个 try 块，然后在 catch 语句中声明 Exception，而不是分离各个可能出现异常的段落并分别捕获其异常。这种做法为分析程序抛出异常的原因带来了困难，因为一大段代码中有太多的地方可能抛出 Exception。

提示：在异常的处理中，尽量减小 try 块的体积。

10.5.6　输出数据不完整

仔细观察代码第 7 行～11 行，不完整的数据是 Java 程序的隐形杀手。仔细观察这段代码，考虑一下如果循环的中间抛出了异常，会发生什么事情。循环的执行当然是要被打断的，其次，catch 代码块会执行，再也没有其他动作了。那么已经输出的数据怎么办？使用这些数据的人或设备将收到一份不完整的（因而也是错误的）数据，却得不到任何有关这份数据是否完整的提示。对于有些系统来说，数据不完整可能比系统停止运行带来的损失更大。较为理想的处置办法是向输出设备写一些信息，声明数据的不完整性。另一种可能有效的办法是，先缓冲要输出的数据，准备好全部数据之后再一次性输出。

根据上面的讨论，下面给出改写后的代码，让读者更清楚的认识异常处理重要性和技巧性，其代码如下：

```
        OutputStreamWriter out=...
        java.sql.Connection conn=...
        try {
          Statement stat=conn.createStatement();
          ResultSet rs=stat.executeQuery(
            "select uid, name from user");
          while (rs.next()) {
              out.println("ID : " + rs.getString("uid") + ", 姓名: " +
rs.getString("name"));
          } }
        catch(SQLException sqlex){
          out.println("警告：数据不完整");
          throw new ApplicationException("读取数据时出现 SQL 错误", sqlex);
        }
        catch(IOException ioex)
         {
          throw new ApplicationException("写入数据时出现 IO 错误", ioex);
```

```
        }
        finally
        {
          if (conn !=null) {
            try {
              conn.close();
            }
            catch(SQLException sqlex2) {
              System.err(this.getClass().getName() + ".mymethod - 不能关闭数据库连
接: " + sqlex2.toString());
            }
          }
          if (out !=null) {
            try {
              out.close();
            }
            catch(IOException ioex2) {
              System.err(this.getClass().getName() + ".mymethod - 不能关闭输出文件
" + ioex2.toString());
            }   } }
```

提示：不要小看这些错误，很多时候即使最有经验的程序员偶尔也会误入歧途，原因很简单，因为它们确确实实带来了“方便”。所有这些反例都可以看做是 Java 编程世界的恶魔，它们美丽动人，无孔不入，时刻诱惑着你。也许有人会认为这些都属于小事，不足挂齿，但请记住，勿以恶小而为之，勿以善小而不为。

10.6　疑难问题解析

本章详细介绍了 Java 异常处理机制、自定义异常等基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：处理异常是一件令程序员十分头疼的事情，对于初学者来说应该如何处理异常呢？

解答：处理异常实际上只有两种方法，一是在发生异常的地方直接处理，二是将异常抛给调用者，让调用者处理。异常处理通常分为检查性异常、运行期异常和错误三种。初学者可从异常的特性入手解决异常。

读者疑问：在进行抛出异常处理时，Java 提供了 throus 和 throw 两种语句，它们之间有什么区别？

解答：throw 语句用在方法体内，表示抛出异常，由方法体内的语句处理，要么和 try…catch 一起使用，要么和 throws 一起使用，但不能单独使用。Throws 语句用在方法声明后面，表示这个方法可能会抛出异常，表示的是一种倾向、可能，但不一定会发生。



职场点拨——不同老板的不同特点

以下属的身份考虑，工作中你最重要的人际关系就是与领导的关系了。在交往过程中，能否真实地领会领导的意图是我们能否处理好与上级的关系的关键因素之一。一定不要随便忽略领导的话，因为在很多时候，领导会故意说一些话来了解真实情况或职员的真实态度。当然领导也有马虎行事、逃避责任的时候，他可能含含糊糊地答应你某件事，而事后却又后悔。究竟怎样判断领导说话是出自内心还是信口开河，这就需要我们下属具备一定的察言观色的能力。在和上级相处时应该掌握如下 4 条技巧。

（1）争取更多与上级交流的机会

不少人错误地认为对上级要敬而远之。俗话说“礼比理更重要”，这个礼首先指的是礼貌，下属应该尽量争取机会与上级交流，让上级熟悉自己、了解自己。而且，上级一般会认为只有那些没有信心或软弱的下属才不敢与其接触。试想一下，上级怎么会把重要的工作交给一个没有自信的下属呢？当然我们也不能过分自信到绝不向上级低头的下属，并且我们需要知道哪些事情能够引起上级的兴趣，哪些是上级欣赏或赞同的事物。

（2）不要排斥上级

很多人一听到上级这个词，心里就不舒服，当初笔者也是这样。一遇见上级发号施令内心就排斥，虽然我们不敢表面表现出这种感觉，但是久而久之在工作中难免会流露出来，到最后就连上级也能感觉到你对他的排斥。而事实上，这种排斥是没有理由的。我们要学会体谅上级，学会站在上级的立场看待问题，并尽量多地看到上级的优点，消除对上级的排斥心理。

（3）避免锋芒毕露

自己锋芒毕露的原因可能是为了得到上级的赏识而拼命表现自己，但是如果比上级还要引人注目就适得其反了。作为下属，一定不要擅自越权，上级非常忌讳下属未经同意就做一些越级的事情，或者是在职权外擅自行动。建议多花一些时间关注一下上级周围的人，包括他的亲属、同学以及能够对他产生影响的人等。这样做对于你是没有任何坏处的，可能会收获意想不到的结果。

（4）突出上级的职位

中国是文明古国，自古就讲究面子、场面，所以身为下属的我们，不妨经常邀请上级出席自己部门或办公室的庆典或其他重要场合，并且抓住机会突出上级的职位，为他争面子。但要在邀请上级出席重要场合之前，一定要确保有把握能控制局面，以免弄巧成拙。

第 11 章　I/O 与文件处理

Java 可以处理计算机中的文件，它为用户提供了 I/O 体系，能帮助用户快速提高数据的处理能力。前面所学的内容都是在内存中处理数据的，无法保存，程序结束时，数据也随之消失。在本章里，将通过 I/O 流对硬盘数据进行读写，以达到长期保存数据的目的。本章主要内容如下：

- ❑ Java I/O 简介。
- ❑ 文件处理。
- ❑ 职场点拨——可以做兼职。

2010 年 XX 月 XX 日，阴

最近我很郁闷，总感觉钱不够花。同学 A 做了一份兼职，几个小时就赚了相当于他一个月工资的钱，我听后很是羡慕，看来我也要寻找一种赚外快的门路了。



一问一答

小菜："我是全职工作人员，能做兼职赚外快吗？"

Wisdom："当然可以，前提是不影响本职工作。你可以利用下班后的时间做兼职，不但能给你带来收入，也可拓展自己的人脉关系。"

小菜："从哪儿获取兼职工作信息呢？"

Wisdom："在 IT 行业，特别是程序员，寻找外快的机会很多，有很多网站上专门介绍程序兼职。在本章的最后，将介绍赚外快的一些途径和方法。"

小菜："赚外快看来确实不错，言归正传，本章将要讲解的 I/O 文件处理很重要吗？"

Wisdom："不管使用什么语言，都难免要接触到文件系统，要经常和文件打交道，不信你可以去书店随便翻一本高级语言的书籍，都会有文件操作的章节。Java 当然也不例外，有人觉得 I/O 的设计很烦琐，其实不然。我倒是觉得只要用熟练了，这个东西还是挺方便的。"

11.1　Java I/O 简介

不管使用什么语言，都离不开对硬盘数据的处理，Java 自然也不例外。那么什么是 I/O 呢？I/O 就是数据输入输出数据流，也称为数据流。下面将详细讲解 I/O 体系的基础知识简

单来说 I/O 就是数据流的输入输出方式。输入模式是由程序创建某个信息来源的数据流，并打开该数据流获取指定信息的数据源，这些数据源都是文件，网络、压缩包或者其他的数据，如图 11−1 所示。

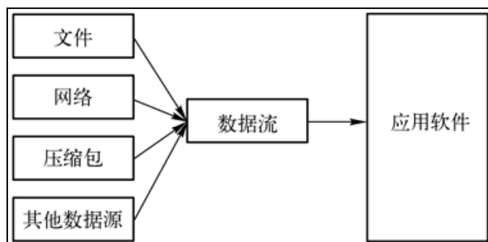


图 11−1　输入模式

输出模式与输入模式恰好相反，输出模式是由程序创建某个输出对象的数据流，并打开数据对象，将数据写入数据流。

11.2　流

流可以分为字节流和字符流，字节流自然以字节为单位对对硬盘数据进行处理，而字符流则是以字符为单位对硬盘数据进行处理的，下面对它们进行讲解。

11.2.1　字节流

通过字节访问和读写数据，可以使用 InputStream 类和 OutputStream 类。这两个类是输入输出的父类，而 FileInputStream 类和 FileOutputStream 类又继承了它，重写了方法。下面进行讲解。

（1）FileInputStream 类

在 Java 程序里，FileInputStream 类下有两个构造方法，介绍如下：

- ❑ FileInputStream（String filepath）。
- ❑ FileInputStream（File fileObj）。

其中 fileObj 要打开的文件，filepath 是文件路径。下面通过一个例子进行讲解。

实例 52：使用 FileInPutStream 类

下面用 FileInputStream 类打开一个文件，其代码见“光盘：源代码/第 11 章/11−1/”：

```
import java.io.*;
public class Liuone {
 public static void main(String args[]){
     try
     {
        //各个方面的异常
         FileInputStream file=new FileInputStream("ai.txt");
         while(file.available()>0)
         {
```

```
                System.out.print((char)file.read());//输出文件内容
            }
            file.close();
        }
                catch(Exception e)
        {
            System.out.println("not found file");
        }
            }
    }
```

运行代码，得到如图 11-2 所示的结果。

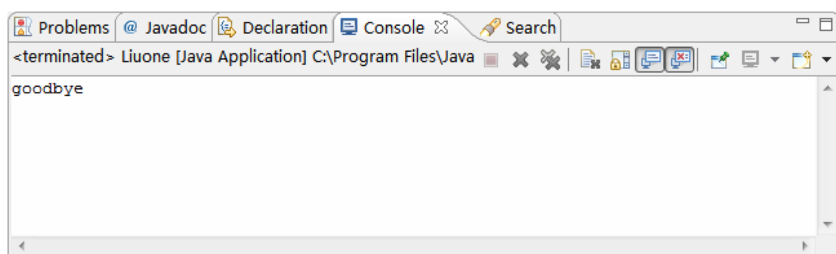
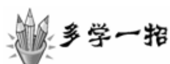


图 11-2　FileInputStream 类



如果是使用 Eclipse 新建的项目，需要将 ai.txt 放置在项目的根目录下，才能执行程序。下面将记事本的字符修改成中文，运行将会得到如图 11-3 所示的结果，试想一下这是什么原因。

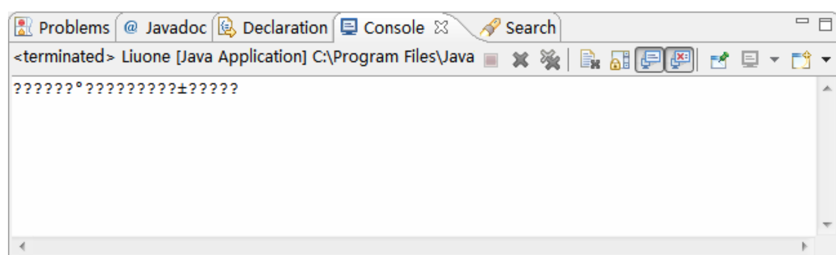


图 11-3　不能显示中文

（2）FileOutputStream 类

FileOutputStream 类的方法与前面所讲解的方法有所不同。下面对 FileOutputStream 类的方法进行介绍：

- FileOutputStream（String filepath）。
- FileOutputStream（File fileObj）。
- FileOutputStream（String filepath，Boolean append）。

其中 filepath 是需要被打开并且需要写入数据的文件名，fileObj 是指被打开并且要输入

的数据文件。

实例 53：使用 FileOutputStream 类

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 11 章/11-2/”：

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class Liuone2 {
 public static void main(String args[]) {
     int i;
     try{
         FileInputStream aa=new FileInputStream("di.txt");
         FileOutputStream bb=new FileOutputStream("do.txt");
         bb.write('0');
         bb.write('7');
         bb.write('.');
         bb.write('8');
         bb.write('.');
         bb.write('1');
         i=aa.read();
         while(i!=-1)
         {
             bb.write(i);
             i=aa.read();
         }
           aa.close();
         bb.close();
     }
             catch(IOException e ){
         System.out.println("do not open this file");
     }
 }
}
```

运行代码，得到如图 11-4 所示的结果。

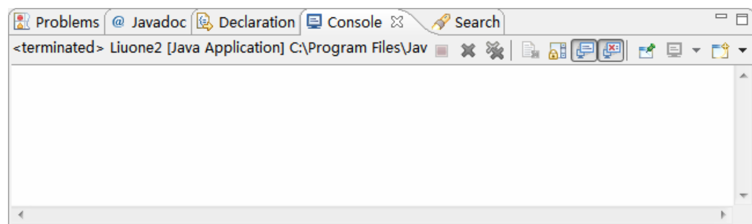


图 11-4　FileOutputStream 类

打开 do.txt 和 di.txt，将 di.txt 文件里的内容复制到了 do.txt 里，如图 11-5 所示。

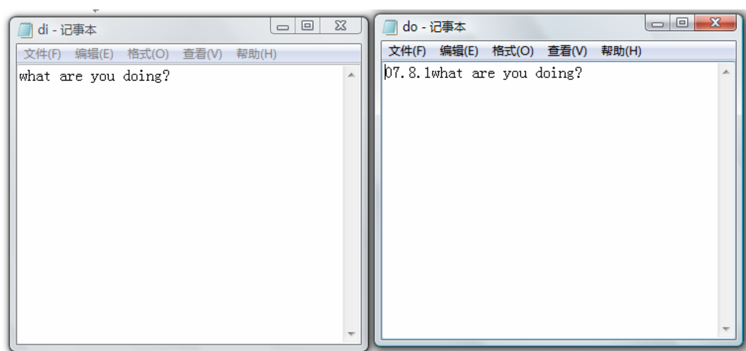


图 11-5　字节流

多学一招

在上面的实例中，讲解了 FileOutputStream 类以及方法，下面再展示一段代码，巩固这一知识点，其代码见“光盘：源代码/第 11 章/11-3/”：

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
class Liuone3
{
 public static void main(String args[])
 {
     int i;
     try
     {
         FileInputStream aa=new FileInputStream("do1.txt");
         FileOutputStream bb=new FileOutputStream("do2.txt");
         bb.write('1');
         bb.write('8');
         bb.write('.');
         bb.write('8');
         bb.write('.');
         bb.write('1');
         i=aa.read();
         while(i!=-1)
         {
             bb.write(i);
             i=aa.read();
         }

         aa.close();
         bb.close();
     }
```

```
                catch(IOException e )
        {
                System.out.println("do not open this file");
        }
        try
    {
                FileInputStream file=new FileInputStream("do1.txt");
                FileInputStream file1=new FileInputStream("do2.txt");
                System.out.println(file);
                System.out.println(file1);
        } catch (FileNotFoundException e) {
                System.out.println("没有寻找到文件");
        }
     }
    }
```

运行代码，得到如图 11-6 所示的结果。

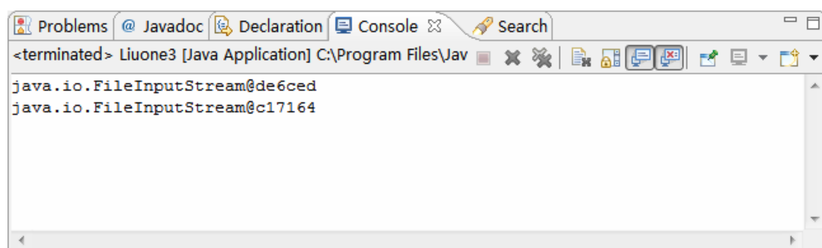


图 11-6　执行结果

打开文件 do1.txt 和 do2.txt 文件，如图 11-7 所示。

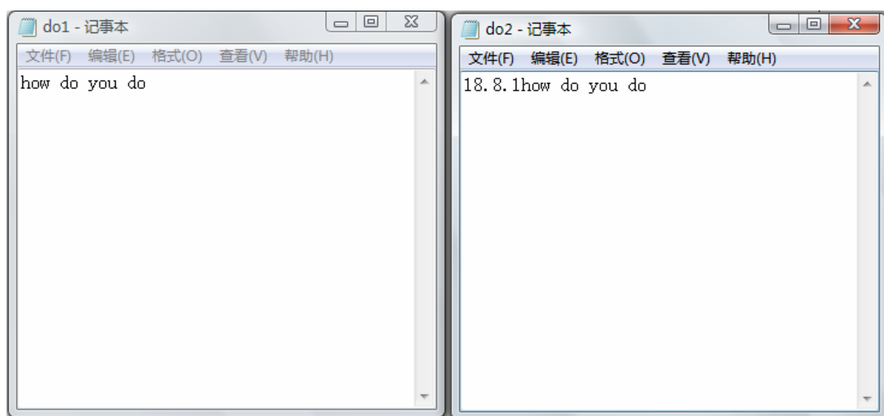


图 11-7　文件里的效果

11.2.2　字符流

字节流是不能处理中文的，但是字符流可以。下面讲解用 FileReader 类和 FileWriter 类

操作字符流。

（1）FileReader 类

在 FileReader 类里，有两个重要的构造方法，介绍如下：

- FileReader (String filepath)。
- FileReader (Fiel fileObj)。

其参数介绍如下：

- Filepath 是文件路径。
- fileObj 是指打开并被读取的文件。

实例 54：使用 FileReader 类

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 11 章/11-4/”：

```
import java.io.*;
public class Liutwo1
{
 public static void main(String args[])
 {

     try
     {
         FileReader fr=new FileReader("1.txt");//使用了 reader 的子类
         char[] c=new char[1];
         while(fr.read(c)!=-1)
         System.out.print(new String(c));
         fr.close();
     }

     catch(Exception e)
     {
         System.out.println(e);
     }
 }
}
```

运行代码，得到如图 11-8 所示的效果。

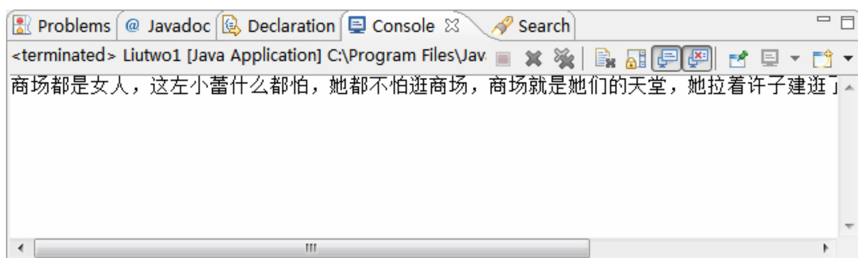
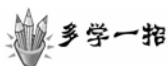


图 11-8　字符流



在上面的实例中，讲解了 FileReader 类以及方法，下面再展示一段代码，巩固这一知识点，其代码见“光盘：源代码/第 11 章/11-5/”：

```
import java.io.IOException;
import java.io.InputStreamReader;
public class Liutwo2{
 public static void main(String[] args) {
     InputStreamReader rin=new InputStreamReader(System.in);
     try {
         char[] cs=new char[50];
         rin.read(cs);
         String str=new String(cs);
         System.out.println("输入的字符串内容：\n\t\t"+str.trim());
     } catch (IOException e) {
         e.printStackTrace();
     }
 }
}
```

运行代码，得到一个空白的结果。

（2）FileWriter 类

FielWriter 类常用的构造方法如下：

- FileWriter (String filepath)。
- FileWriter (String filepath，Boolean append)。
- FileWriter (String fileObj)。

其参数介绍如下：

- filepath 是文件路径。
- fileObj 是指打开并被读取的文件。
- Boolean append 这个参数尤为重要，如果为 True，它将以追加的方式打开，如果为 False，则以覆盖的方式打开。

实例 55：使用 FileWriter 类

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 11 章/11-6/”：

```
import java.io.*;
public class Liutwo3
{
 public static void main(String args[]) throws IOException
 {
     int i;
     FileReader fr;
     FileWriter fw;
     try
     {
```

```
            fr=new FileReader("1.txt");
        }
        catch(FileNotFoundException e)
        {
            System.out.println("not found this file");
            return;
        }
        try
        {
            fw=new FileWriter("2.txt");
        }
        catch(FileNotFoundException e)
        {
            System.out.println("error");
            return;
        }
        try
        {
            i=fr.read();
            while(i!=-1)
            {
                fw.write(i);
                i=fr.read();
            }

                fr.close();
                fw.close();
            }
            catch(IOException e)
            {
                System.out.println("error");
            }
    }
  }
```

运行代码，得到如图 11-9 所示的结果。

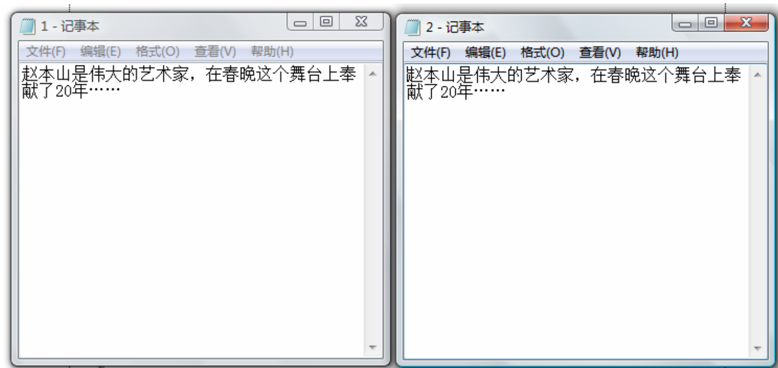
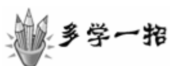


图 11-9　字符流执行结果



在上面的一个实例中，讲解 FileWriter 类以及方法，下面再展示一段代码，巩固这一知识点，其代码见“光盘：源代码/第 11 章/11-7/”：

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
public class Liutwo4{
 public static void main(String args[]) throws IOException    {
     int a;
     FileReader de;
     FileWriter fw;
     try
     {
         de=new FileReader("wei1.txt");
     }
     catch(FileNotFoundException e)
     {
         System.out.println("not found this file");
         return;
     }
     try
     {
         fw=new FileWriter("wei2.txt");
     }
     catch(FileNotFoundException e)
     {
         System.out.println("error");
         return;
     }
     try
     {
         a=de.read();
         while(a!=-1)
         {
               fw.write(a);
               a=de.read();
         }

         de.close();
         fw.close();
     }
     catch(IOException e)
     {
```

```
                System.out.println("error");
            }
        }
    }
```

运行代码，得到如图 11-10 所示的结果。

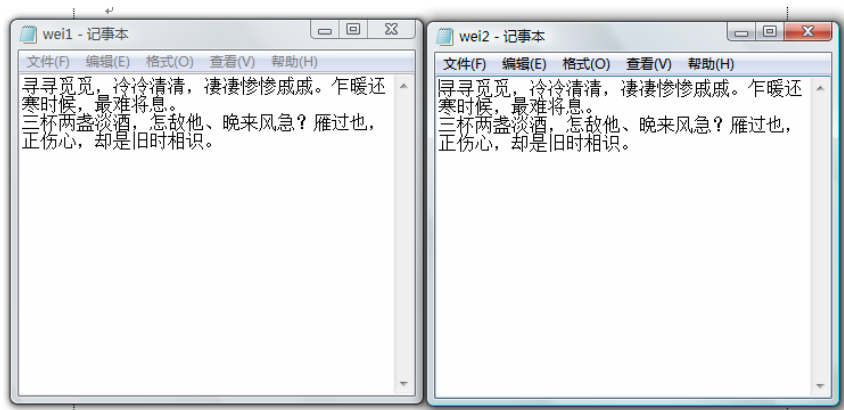


图 11-10　执行结果

11.3　加快 I/O 操作效率

上一节讲解了字节流和字符流，但是通过流进行操作的效率很低，不适合远程操作或大型的项目，如何加快它们的处理速率呢？下面对它进行讲解。

11.3.1　缓冲字节流

加快字节流用户可以使用缓冲数据流进行处理，在前面讲解了对硬盘的读写使文件流的效率太低，在这一节将讲解 BufferedInputStream 类和 BufferedOutputStream 类。当输入数据时，数据以块为单位读入缓冲区。它有下面的构造方法：

- BufferedInputStream (Obj)。
- BufferedOutputStream (Obj)。

其中 Obj 为已经建立好的输入/输出数据流的对象。

实例 56：使用 BufferedInputStream 类和 BufferedOutputStream

下面将给出一段程序，讲解 BufferedInputStream 类和 BufferedOutputStream 类，其代码见“光盘：源代码/第 11 章/11-8/”：

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
```

```
public class Huanc
{
 public static void main(String args[]) throws IOException
 {
     FileInputStream fis;
     FileOutputStream fos;
     BufferedInputStream bis;
     BufferedOutputStream bos;

     int i;
     try
     {
         fis=new FileInputStream("wei1.txt");
         bis=new BufferedInputStream(fis);
         fos=new FileOutputStream("wei2.txt");
         bos=new BufferedOutputStream(fos);

         i=bis.read();
         while(i!=-1)
         {
             bos.write(i);
             bos.flush();
             i=bis.read();
         }

         fis.close();
         fos.close();
         bis.close();
         bos.close();
         System.out.print("复制成功");
     }

     catch(IOException e)
     {
         System.out.println("没有找到文件");
     }
 }
}
```

运行代码，得到如图 11-11 所示的结果。

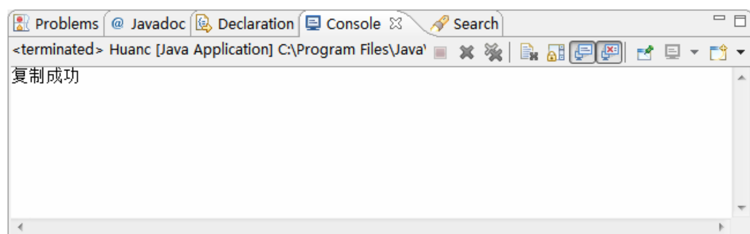


图 11-11　复制成功

打开 weil.txt 和 wei2.txt，将 wei1.txt 里的内容复制到 wei2.txt 文件里，其结果如图 11-12 所示。

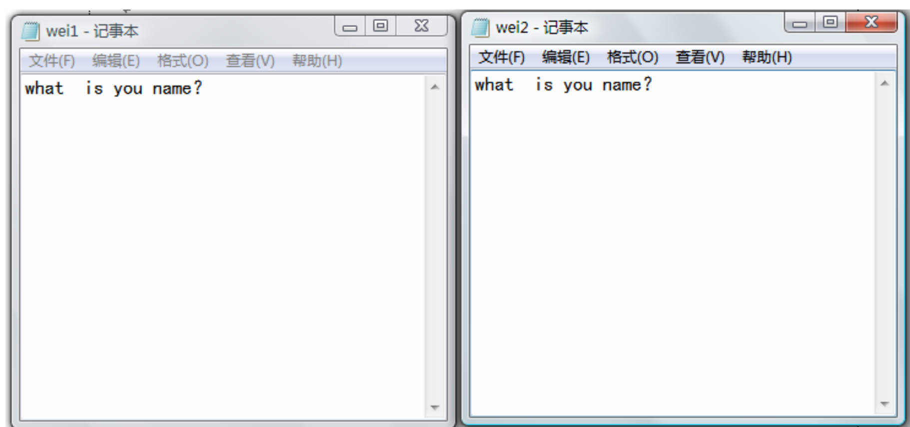


图 11-12 复制文件

多学一招

在上面的实例中，讲解了 BufferedInputStream 类和 BufferedOutputStream 类，快速地复制了文件。下面再展示一段代码，巩固这一知识点，其代码见“光盘：源代码/第 11 章/11-9/”：

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class Huancc
{
 public static void main(String args[]) throws IOException
 {
    FileInputStream bu1;
    FileOutputStream bu2;
    BufferedInputStream bu3;
    BufferedOutputStream bu4;
    int i;
    try
    {
        bu1=new FileInputStream("wei1.txt");
        bu3=new BufferedInputStream(bu1);
        bu2=new FileOutputStream("wei2.txt");
        bu4=new BufferedOutputStream(bu2);
               i=bu3.read();
        while(i!=-1)
```

```
            {
                bu4.write(i);
                bu4.flush();
                i=bu3.read();
            }
           bu1.close();
           bu2.close();
           bu3.close();
           bu4.close();
        }
        catch(IOException e)
        {
           System.out.println("没有找到文件");
        }
    }
}
```

运行代码，得到如图 11-13 所示的结果。

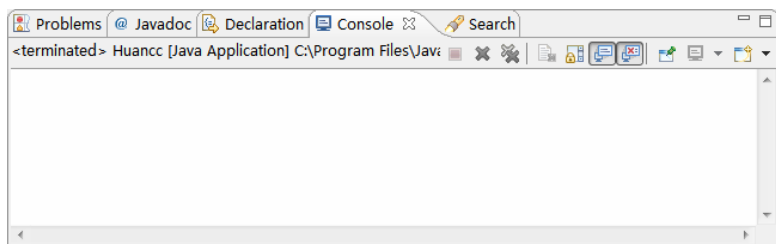


图 11-13　执行结果

打开两个文件，进行复制，得到如图 11-14 所示的结果。

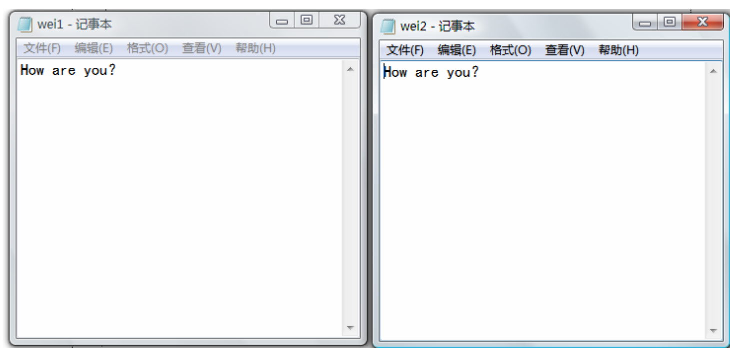


图 11-14　缓冲数据流

11.3.2　缓冲字符流

字符处理也可以和字节一样，通过缓冲的方式，加快处理效率。Java 为用户提供了 BufferedReader 类和 BufferedWrote 类，它和缓冲字节流一样，引进缓冲字符流是为了提供

I/O 流的操作速度。BufferedReader、BufferedWroter 类通常使用的构造方法介绍如下：

- ❑ BufferedReader（Obj）。
- ❑ BufferedWroter（Obj）。

实例 57：使用 BufferedReader 类和 BufferedWrote 类

下面将给出一段程序，讲解 BufferedReader 类和 BufferedWrote 类，其代码见“光盘：源代码/第 11 章/11-10/”：

```
import java.io.*;
public class huand
{
 public static void main(String args[])
 {
     try
     {
         FileReader fr=new FileReader("cq.txt");
         BufferedReader br=new BufferedReader(fr);
         String a;
         while((a=br.readLine())!=null)
         {
            System.out.println(a);
         }
         fr.close();
         br.close();
     }
     catch(IOException e)
     {
         System.out.println(e);
     }
         }
}
```

运行代码，得到如图 11-15 所示的结果。

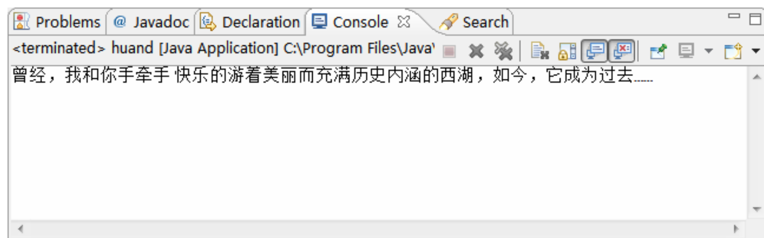
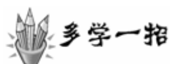


图 11-15　缓冲字符流



字节流是以单个字节为单位的，而字符流是以单个字符为单位，通常情况下，一般的字符是由多个字符组成的。Java 语言内部可将一个字节转换为字符流，这种转换的工作是由

InputStreamReader 和 OutputSteamWriter 完成的，它们各自有两个构造方法，通过这些方法，可以相互转换，得到符合要求的输入/输出流。

11.4　文件处理

文件处理方式没有字节流和字符流使用频繁，但是它对文件的操作是最直接和最详细的，通过它可以获取很多的指定文件信息。

11.4.1　文件类

如果要对一个文件进行操作，首先需要知道这个文件的基本信息，文件类可以提供获取这些信息的方法，下面讲解 File 的方法，其结构如下：

- File（String path）。
- File（String path，String file）。
- File（File Obj，String file）。

参数介绍如下：

path 是文件的路径名。

file 是文件名。

obj 是 File 的对象。

下面介绍它的一般方法：

- Boolean exists()：测试 File 对象对应的文件是否存在。
- Boolean isFile ()：测试 File 对象对应的是否是一个文件。
- Boolean isDirectory()：测试 File 对象对应的是否是一个目录。
- Boolean canRead()：测试 File 对象对应的文件是否可读。
- Boolean canWriter()：测试 File 对象对应的文件是否可写。
- Boolean isHidden()：测试 File 对象对应的文件是否隐藏。
- Boolean createNewFile()：创建一个新文件。
- Boolean setReadOnly()：将 File 文件对象对应的文件设置刻度。
- String[] list()：获取 File 对象对应的文件目录。
- Boolean Mkdir ()：创建 File 对象对应的文件目录。
- Boolean delete()：删除 File 对象对应的文件。
- Boolean equals(Object obj)：比较两个文件对象。
- Boolean renameTo(File dest)：将对应的对象重命名。
- String toString()：将 File 对象转换为字符串。
- String getString()：返回 File 对象对应的文件的名。
- String getParent()：返回 File 对象对应的文件的父目录
- String getPath()：返回 File 对象对应的文件的绝对路径。
- String AbsolutePath()：返回 File 对象对应的文件的绝对路径。
- Long lastModified：返回文件最后修改的时间。
- Long length（）：返回文件的大小。

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 11 章/11-11/”：

```
import java.io.*;
class FileOne
{
 public static void main(String args[])
 {
     File a=new File("D:\\图片");
     File b=new File("D:\\图片","新世纪.doc");
     File c=new File(a,"新世纪.doc");
     if(a.exists())
     {
         System.out.println("a 检查到 D 盘有图片文件夹");
     }

     if(b.isFile())
     {
         System.out.println("b 检查到 D 盘的图片文件夹有新世纪.doc 文件");
     }
     if(c.isFile())
     {
         System.out.println("c 检查到 D 盘的图片文件夹有新世纪.doc 文件");
     }
 }
}
```

执行程序前，注意要将图片文件夹复制到 D 盘根目录下，然后再运行，最后得到如图 11-16 所示的结果。

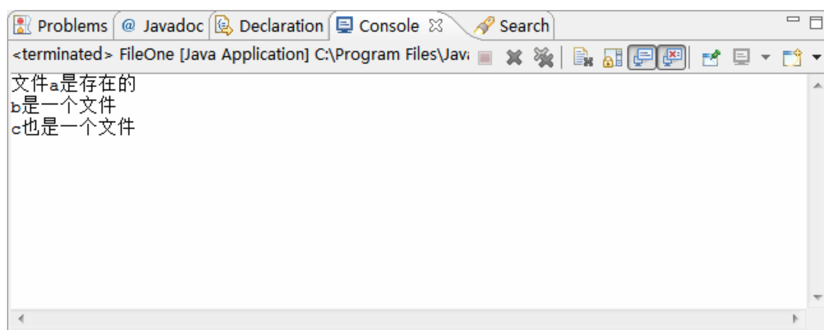


图 11-16　文件类

11.4.2　使用文件类处理文件

使用文件类处理文件是十分常见的，为了加深读者对它的理解，下面通过一段代码进行讲解，其代码见“代码 48：光盘：源代码/第 11 章/11-12/”：

```
import java.io.File;
import java.io.IOException;
public class Filetwo
{
 public static void main(String args[])
 {
     File f1=new File("D:\\mother\\wei");
     File f2=new File(f1,"1.txt");
     File f3=new File("E:\\father\\1.txt");
        System.out.println(f2);
        System.out.println();
        System.out.println("exists:        "+f2.exists());
     System.out.println("isfile:        "+f2.isFile());
     System.out.println("Directory:     "+f2.isDirectory ());
     System.out.println("Read:          "+f2.canRead());
     System.out.println("Write:         "+f2.canWrite());
     System.out.println("Hidden:        "+f2.isHidden());
     System.out.println("ReadOnly:      "+f2.setReadOnly
());
     System.out.println("Name:          "+f2.getName());
     System.out.println("Parent:        "+f2.getParent());
     System.out.println("AbsolutePath:    "+f2.getAbsoluteFile());
     System.out.println("lastModified:   "+f2.lastModified  ());
     System.out.println("length:        "+f2.length());
     System.out.println("list           "+f2.list());
     System.out.println("mkdir          "+f2.mkdir());
     File f4=new File("Student.java");
     System.out.println("newname"+f2);
     f2.renameTo(f4);
     System.out.println("Name:          "+f4.getName());
     System.out.println(f2+"exist       "+f2.exists());
     System.out.println(f4+"exist       "+f2.exists());
     System.out.println("delete"+f4);
     System.out.println("equals         "+f2.equals(f4));
     f4.delete();
     System.out.println(f4+"exist       "+f4.exists());
     System.out.println("String         "+f2.toString());
 }
}
```

将光盘中的 father 和 mother 文件复制到 D 盘根目录下，然后运行代码，得到如图 11-17 所示的结果。

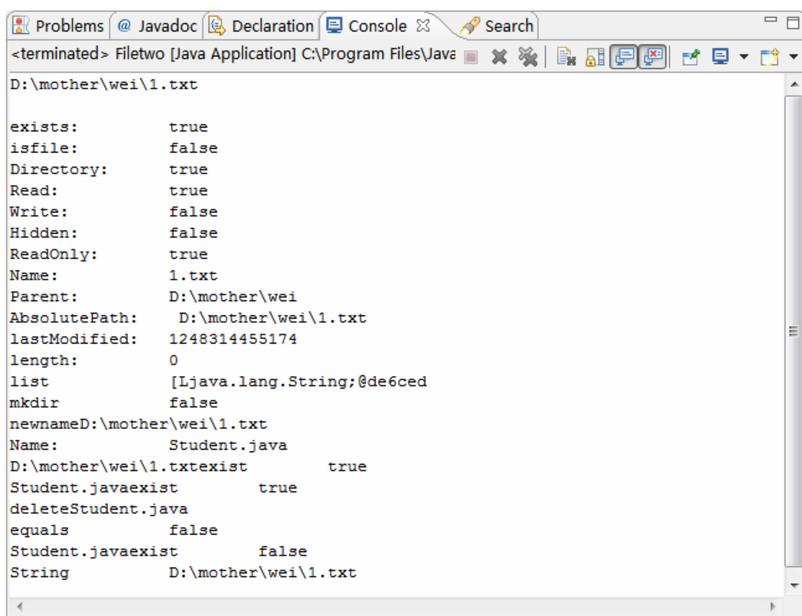


图 11-17 执行结果

11.5 疑难问题解析

本章详细介绍了 I/O 简介、文件处理等基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：在前面的一些章节中，讲解了字符流和字节流，它们之间有什么联系和区别？

解答：有一定计算机基础的人都知道字符和字节，数字、英文等字符都可以使用字节表示，但是汉字不能以字节表示，因为一个汉字需要多个字节才能表示，所以说很明显字符流主要是针对汉字的，而字节流是用来处理字母、数字等内容的。

读者疑问：缓冲字节流和缓冲字符流，其主要目的是加快程序对文件的处理，它和前面讲解的字节流和字符流，有什么区别？

解答：最大的区别是速度，缓冲字节流和缓冲字符流从速度上看要比字节流和字符流要快许多，不过对于少量的文件处理，是很难发现处理速度的快慢的。



职场点拨——可以做兼职

（1）获得兼职的途径

在当今生活中，几乎每个人都面临着一个很严峻的问题：钱！无论是买房、买车、旅行散心，还是和三五知己一起聚会，都得需要钱。程序员也是凡夫俗子，也得赚钱来养家糊口。固

定的那些工资有时满足不了我们花钱的欲望，怎么办？兼职是一个很好的解决问题的途径。现在 IT 行业外包盛行，很适合发展自己的兼职业务。程序员可以通过以下渠道获取额外收入。

1）客户：你曾经为这个客户服务过，只要他认可你的技术，你就有很大机会从他手上接私活。因为以前曾经合作过，所以在流程上会很顺利。而他认识你的同事或上司，可能不经意间泄露你们的合作关系，对你的本职工作会有影响。

2）朋友介绍：日常生活中，朋友和同学都可能会为你介绍客户。通过朋友介绍找客户的优点是对方肯定真实可靠，缺点是可能碍于面子，收入不会很高，而且日后服务的过程中可能会耗费较多的时间。

3）网络资源：当前因特网迅猛发展，各个私活站点如雨后春笋般纷纷建立。我们可以登录一些主流网站的私活板块，来寻找自己的私活。例如威客、365huo、CSDN 等，都是主流的私活交易平台。利用网络资源的优点是信息量大，缺点是竞争压力大，信任度较低。

4）开一个工作室：若时间充裕，可以叫上三五个知己或同学，合伙开一个工作室。当今网络普及，能够很好地完成洽谈和交流等工作。加上交通便利发达，可以利用周末时间去外地洽谈业务，而不影响正常工作。

（2）确保合作成功

但是即使找到了客户，你确定能顺利合作成功吗？在此笔者有以下几个建议供读者参考：

1）都说朋友多了路好走，现实还真是这样。我们在寻找兼职时，最好多接受朋友或熟人推荐的私活，这样不但双方都比较放心，而且一般也不会欠款。反之如果是陌生人的话，项目难以拿到不说，尾款的结算速度也会很慢。

2）对于 Web 项目来说，美工和程序是分开的。在接活前一定先跟美工把报酬讲好，如果程序员和美工报酬一样的话，建议不要接，这不是刻意抬高我们程序员，而是因为后期工作主要是功能改动和项目升级，程序员的工作占绝大多数，而美工的任务就少得多。

3）在开始项目洽谈的时候，一定让客户把需求写成书面形式，然后根据文本里的要求进行估价。一方面我们可以了解整个工程量，另一方面也能作为要价的根据。

4）要跟客户挑明从项目完成并结账的那天起，我们就不负责了。除非和客户事先商定好进行免费维护或有偿维护。

5）项目的后期维护是很花费精力和时间的，尽量让客户提供资金支持。

6）特别对于 Web 项目来说，程序和页面一定要独立实现，这样做的好处是不易产生误会，最主要的是能提高工作效率，后期美工修改起来也不会影响到程序。

7）在具体开工前一定要清楚报酬到底是多少，究竟是税前的还是税后的，因为很多单位在结账时会扣掉个人所得税。

8）最好有自己的服务器，可以把做的项目放到服务器上。这样能向客户展示我们的作品，展示项目进度。如果客户满意了，等对方给我们结款之后再把代码给他们，会避免出力而一无所获的局面。

9）包装自己：我们需要包装自己的技术和作品。可以将以前做过的案子都放在网上，例如放在 QQ 空间中就是一个很好的选择。

10）在做一个项目时，无论是 Web 项目还是桌面项目，建议把整个项目分成几段。例如可以把网站分为“美工”、“后台”、“美工和后台”的结合，每做完一期，客户满意就给钱，这样能降低风险。

第 12 章　Java 线程

在开发程序的时候，一定会结束线程。前面讲解过的程序都是单线程，那么什么是多线程呢？程序需要同时处理多项任务的时候，就需要多线程开发，读者肯定都知道，若一个程序在同一时间只能做一件事，其功能就会显得过于简单，就肯定无法满足现实的需求。所以说在实际应用中，多线程开发是必不可少的。本章将详细讲解多线程的开发。本章主要内容如下：

- 线程基础。
- 创建线程。
- 创建多线程。
- 线程的优先级。
- 多线程同步。
- 让线程之间互相通信。
- 职场点拨——领会老板的话。

2010 年 XX 月 X 日，阴

今天中午，主管对我说："小菜，明天下班后晚走一会儿，帮我捎个东西"。我很是纳闷，主管到底是怎样想的呢？是器重我？还是看着我不干活，特意下班后要给我洗脑？



一问一答

小菜："今天领导的安排让我摸不着头脑，不知是话里有话，还是我多想了？"

Wisdom："在职场中，作为一个职员总会这样或那样听到领导的善意关怀，也可能会接到温馨的任务安排。倘若你想在职场中有所提升，就必须彻底明白领导的真正含义，这样才能确保自己不会因领会错误而做错事，影响工作效率。"

小菜："先不说这些了，言归正传 Java 是怎样提高自身的效率的呢？"

Wisdom："提到 Java 的工作效率，我们就不得不说多线程，多线程的功能是确保程序理解服务器的真正含义，是为提高效率而生的。

12.1　线程起步

不管是在 C、C++，或者其他的程序设计语言中，线程都是十分重要的知识点。多线程是现代软件系统开发的发展方向，Java 作为当今的主流程序设计，自然是支持多线程的。多

线程具有并发性，其执行的效率非常高。

12.1.1　线程与进程的理解

每个正在系统上运行的程序都是一个进程。每个进程包含一到多个线程。进程也可能是整个程序或者是部分程序的动态执行。而线程则是一组指令的集合，或者是程序的特殊段，它可以在程序里独立执行。也可以把它理解为代码运行的上下文。所以线程基本上都是轻量级的进程，它负责在单个程序里执行多任务。通常由操作系统负责多个线程的调度和执行，打开任务管理器，就可以看到多个线程，如图 12-1 所示为系统里的进程。

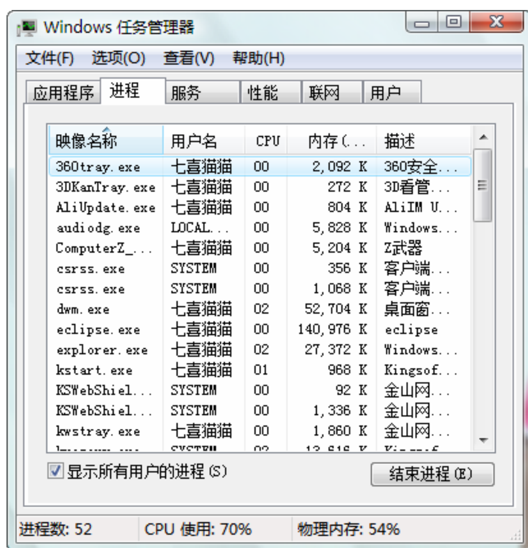


图 12-1　系统里的进程

12.1.2　多线程的理解

多线程可以使多个线程并行，以同时完成多项任务，提高系统的效率。使用线程的好处有以下几点：

- 使用线程可以把占据较长时间的程序和任务放到后台去处理。
- 用户界面可以更加吸引人，比如用户点击了一个按钮去触发某些事件的处理，可以弹出一个进度条来显示处理的进度。
- 程序的运行速度可能加快。
- 在等待任务完成时实现让用户输入、文件读/写和网络收发数据等。线程在这种情况下可以释放一些珍贵的资源，比如内存占用等。

Java 的运行系统在很多方面都是基于线程的，所有的类设计都考虑到多线程，和别的编程语言相比，它是十分优秀的。Java 让 Java 程序员能够使用并发编程机制。下面通过一个简单例子，简单介绍多线程，其代码如下：

```
public class XianChen
{
```

```
Public static void main(String args[])
{
System.out.println("单线程");
}
}
```

其中 main（）方法就是一个多线程，main 函数在程序运行的时候由 JVM 负责启动。

12.2 创建线程

创建线程十分简单，用户只需要将创建的类实现 Runable 接口和继承 Thead 类。在本节中将主要学习创建多线程，以及常用到的多线程方法和多线程思想。

12.2.1 创建主线程

什么是主线程呢？主线程就是在程序启动时立刻执行，它是所有现场最早运行的线程。下面通过一个实例进行讲解。

实例 58：使用程序自动创建主线程

下面将给出一段程序，讲解程序自动创建主线程，其代码见“光盘：源代码/第 12 章/11-10/”：

```
public class Xianzhu {
 public static void main(String args[])
 {
     //获得主线程的引用
     Thread t=Thread.currentThread();
     //获得主线程信息
     System.out.println("主线程:"+t);
     //改变主线程的信息
     t.setName("My Thread");
     //主线程的新信息
     System.out.println("改变名称后:"+t);
     //线程内容
     try
     {
         for(int i=0;i<10;i++)
         {
             System.out.println(i);
             //每隔 2 秒打印一个数
             t.sleep(2000);
         }
     }
     catch(Exception e)
     {
         System.out.println("发生异常");
     }
```

```
        }
    }
```

运行代码，得到如图 12-2 所示的结果。

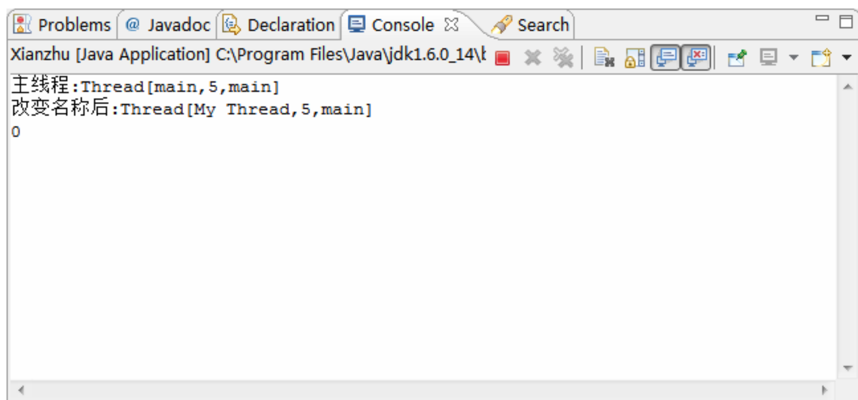


图 12-2　创建主线程

等待 20 秒，得到如图 12-3 所示的结果。

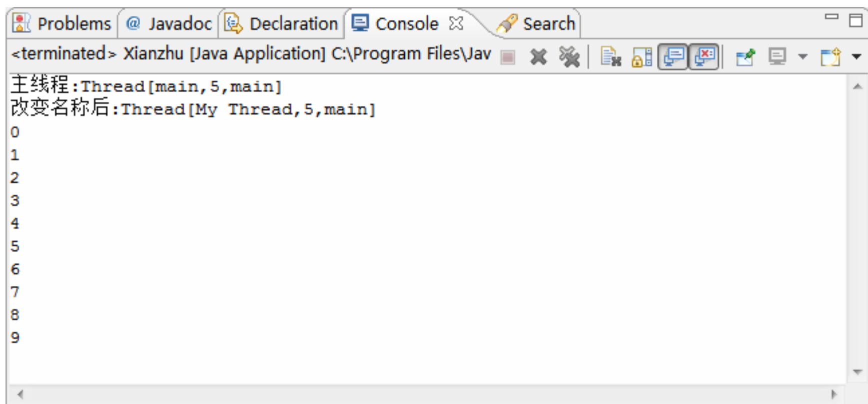


图 12-3　创建主线程执行完成的结果

多学一招

在主线程的应用程序中，通过调用 currentThread（）获得主线程的引用，用局部变量 t 来表示该引用，最好编写一些线程内容，然后每隔两秒打印一个数。

12.2.2　通过 runable 接口创建线程

让类通过实现 runable 接口，即可创建多线程，其格式如下：

```
Public class simple implements Runnable
{
//实例化对象
```

```
Simple simple=new Simple();
Thread start();
}
```

参数介绍如下：

- Simple：类名称。
- Implements：实现接口语句。
- Simple：实例化 Simple 对象。

下面将给出一段程序，创建一个类，通过实现 runable 接口创建线程，其代码见“光盘：源代码/第 12 章/Duoone1.java”：

```
public class Duoone1
 {
 public static void main(String args[])
 {
     //实例化一个线程
     MyThread r=new MyThread();
     Thread t=new Thread(r);
     //对 run 方法进行调用
     t.start();
     //主程序内容
     for(int x=0;x<6;x++)
     {
         System.out.println("大");
     }
 }
}
//通过实现 Runnable 接口创建线程
class MyThread implements Runnable
{
 int y;
 //定义 run 方法
 public void run()
 {
     //线程内容
     while(true)
     {
         System.out.println("小"+y++);
         if (y==6)
         {
             break;
         }
     }
 }
}
```

运行代码，得到如图 12-4 所示的结果。

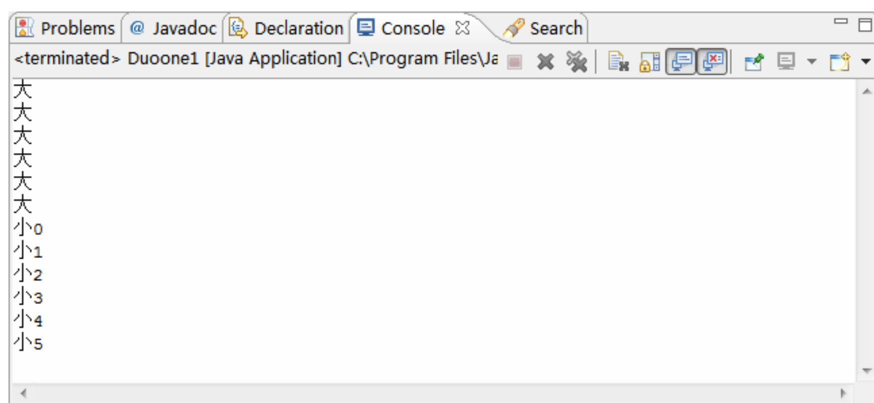


图 12-4　创建多线程

12.2.3　通过 Thread 类创建线程

在编程的过程中，用户不但可以用 Runnable 接口创建线程，还可以通过继承 Thread 创建线程，创建的方法和上面的接口大致相同，其介绍如下：

- Thread()。
- Thread(Runnable target)。
- Thread(Runnable target,String name)。
- Thread(String name)。
- Thread(ThreadGroup group,Runnable target)。
- Thread(ThreadGroup group,Runnable target,Sting name)。
- Thread(ThreadGroup group, Sting name)。

下面创建一段代码，通过 thread 类创建线程，让程序的结果如上一节所列程序一样，其代码见“光盘：源代码/第 12 章/Duoone2.java”：

```
public class Duoone2
 {
 int x;
 public static void main(String args[])
 {
     //实例化一个线程
     Myone t=new Myone();
     //对 run 方法进行调用
     t.start();
     //主程序内容
     for(int x=0;x<6;x++)
     {
         System.out.println("大");
     }
 }
```

```
    }

class Myone extends Thread
{
 int y;
 //定义 run 方法
 public void run()
 {
     //线程内容
     while(true)
     {
         System.out.println("小"+y++);
         if(y==6)
         {
             break;
         }
     }
 }
}
```

运行代码，得到如图 12-5 所示的结果。

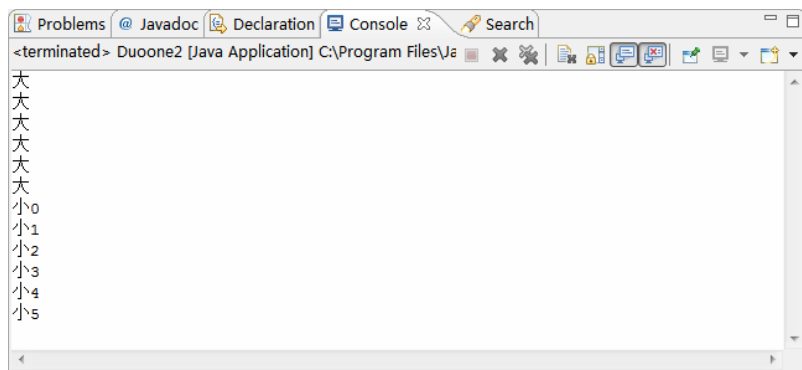


图 12-5　创建线程

12.3　创建多线程

前面已经讲过，可以通过实现 Runable 接口和继承 Thread 类创建多线程，从面向对象的角度看，Thread 类是一个虚拟处理机严格的封装，因此只有当处理机模型修改或扩展时，才能继承类。Java 程序设计中，类是单一继承，如果一个类继承 Thread 类，就不能继承其他的类，所以用户只有通过实现接口来完成多线程的程序功能，类中 this 指向的实际是 Thread 类的实例。下面通过一段代码来讲解多线程，其代码见“光盘：源代码/第 12 章/Duotwo.java”：

```
public class Duotwo{
```

```
    public static void main(String args[])
   {
        //创建三个线程
        MyThread1 mt1=new MyThread1("No.1 线程");
        MyThread2 mt2=new MyThread2("No.2 线程");
        MyThread3 mt3=new MyThread3("No.3 线程");
        mt1.start();
        mt2.start();
        mt3.start();
    }
   }
   class MyThread1 extends Thread
   {
    String name;
    MyThread1(String threadname)
    {
        name=threadname;
    }
    public void run(){
        try
        {
            for(int i=0;i<5;i++)
            {
                System.out.println(name+":"+i);
                Thread.sleep(1000);
            }
        }
        catch(Exception e)
        {
            System.out.println(name+"异常");
        }
        System.out.println(name+"退出");
    }
   }
   class MyThread2 extends Thread
   {
    String name;
    MyThread2(String threadname)
    {
        name=threadname;
    }
    public void run()
    {
        try
        {
            for(int i=0;i<5;i++){
```

```
                System.out.println(name+":"+i);
                Thread.sleep(1000);
            }
    }
        catch(Exception e){
            System.out.println(name+"异常");
        }
        System.out.println(name+"退出");
     }
    }
    class MyThread3 extends Thread
    {
     String name;
     MyThread3(String threadname)
     {
        name=threadname;
     }
     public void run()
     {
        try
        {
            for(int i=0;i<5;i++)
            {
                System.out.println(name+":"+i);
                Thread.sleep(1000);
            }
        }
        catch(Exception e)
    {
            System.out.println(name+"异常");
        }
        System.out.println(name+"退出");
     }
    }
```

运行代码，得到如图 12-6 所示结果。

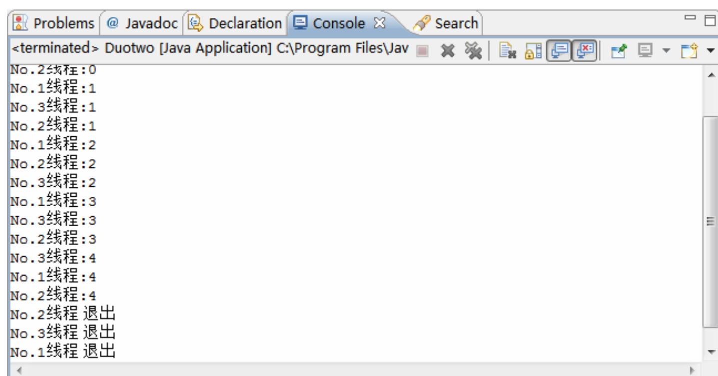


图 12-6　创建多线程

12.4　线程的优先级

线程和运算一样，有自己的优先级。线程优先级是用来判断何时执行线程程序，从理论上讲，线程优先级高的线程会比优先级低的线程获得更多的 CPU 时间，获得 CPU 的时间长短与很多因素有关。设置线程的优先级可以使用 setPriority（）方法，其格式如下：

```
Final void setPriority（int level）
```

参数介绍如下：

- Level 是设置线程的优先级的大小。
- MIN-PRIORITY 表示优先级最小，即 1。
- MAX-PRIORITY 表示优先级最大。
- 优先级的大小不能超过 10。

获得当前线程的优先级也十分简单，其代码如下：

```
Final getPriotity()
```

实例 59：使用循环记录次数来表现出优先级的高低

下面将给出一段程序，它通过循环，记录次数来表现出优先级的高低，其代码见“光盘：源代码/第 12 章/you.java”：

```
class duoyou implements Runnable
{
 int click=0;
 Thread t;
 //应用关键字 volatile
 private volatile boolean running = true;
 public duoyou(int p)
 {
     t=new Thread(this);
     t.setPriority(p);
 }
 public void run()
 {
     while (running)
     {
         click++;
     }
 }
 public void stop()
 {
     running=false;
 }
 public void start()
```

```
        {
            t.start();
        }
    }

public class you
{
 public static void main(String args[])
 {
     Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
     duoyou hi=new duoyou(Thread.NORM_PRIORITY+2);
     duoyou lo=new duoyou(Thread.NORM_PRIORITY-2);
     hi.start();
     lo.start();
     try
     {
         Thread.sleep(10000);
     }
     catch(Exception e)
     {
         System.out.println("主线程异常");
     }
     lo.stop();
     hi.stop();
     try
     {
         hi.t.join();
         lo.t.join();
     }
     catch(Exception e)
     {
         System.out.println("异常");
     }
     System.out.println("低优先级线程:"+lo.click);
     System.out.println("高优先级线程:"+hi.click);
 }
}
```

运行代码，得到如图 12-7 所示的结果。

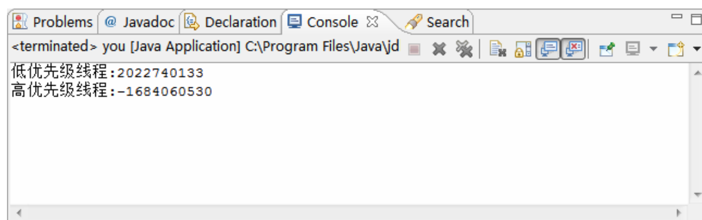
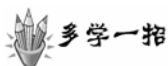


图 12-7　线程的优先级



上面的输出程序与自己计算机的 CPU 有关，不同的计算机有着不同的结果，这取决于 CPU 处理的速度，在同一台计算机上运行，也会出现不同的效果。下面通过一段程序进行讲解，其代码见“光盘：源代码/第 12 章/youone.java”：

```
import java.io.*;
public class Youone extends Thread
{
 public Youone(int priority,String name)
{
     setPriority(priority);
     setName(name);
     start();
 }
 public String toString()
{
     return super.toString()+":"+getName();
 }
 public void run()
{
     int countdown=5;
     while(true){
         System.out.println(this);
         if(--countdown==0)return;
     }
 }
 public static void main(String[] args)
 {
     for(int i=0;i<5;i++){
         if(i==3){
             new Youone(Thread.MAX_PRIORITY,"最高优先级线程-"+i);
             continue;
         }
         new Youone(Thread.NORM_PRIORITY,"低优先级线程-"+i);
     }
     new Youone(8,"高优先级线程");
 }
}
```

运行代码，得到如图 12-8 所示的结果。

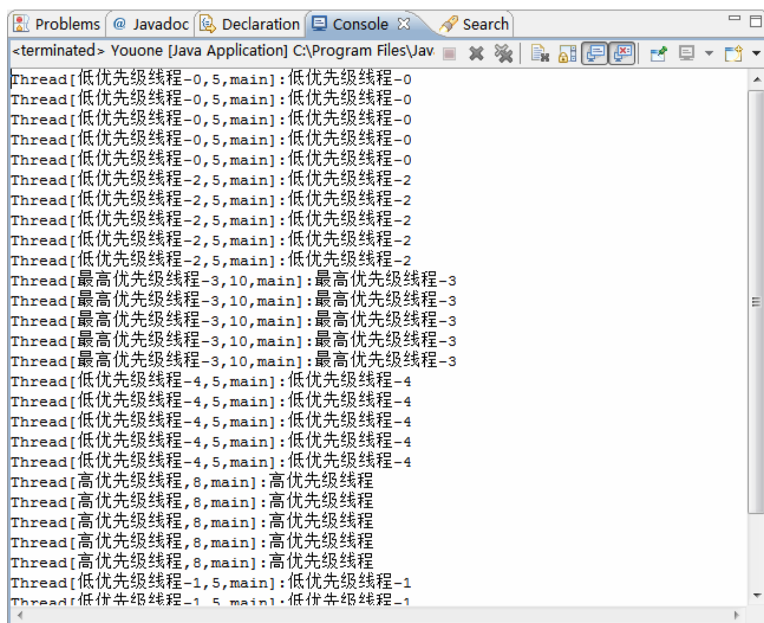


图 12-8　线程的优先级

12.5　控制线程

线程有自己的优先级，但是当遇到如有的线程临时调到前面去，有的线程需要临时中断之类的情况应该怎么办呢？本节将讲解控制线程。

12.5.1　当前的线程等待

在线程中使用方法 join()可以让当前线程暂时停止，直到 join()方法所调用的那个线程结束，再继续执行。

实例 60：用方法 join()线程进行等待

下面将给出一段程序，用方法 join()线程等待，其代码见“光盘：源代码/第 12 章/Deng.java”：

```
class xiancheng extends Thread{
 public void run()
 {
     for(int i=0;i<20;i++)
     {
         System.out.print("o");
         try
         {
             Thread.sleep(1000);
         }
         catch(InterruptedException e)
```

```
            {
                System.out.println("异常"+e);
            }
        }
    }
}
class xiancheng2 extends Thread{
 public xiancheng mt;
 public void run() {
      for(int i=0;i<20;i++)
      {
            System.out.print("※");
            try
            {
                Thread.sleep(1000);
            }
            catch(InterruptedException e)
            {
                System.out.println("异常"+e);
            }
            if(i==10)
            {
                try
                {
                     mt.join();
                 }
                 catch(Exception e)
                 {
                      System.out.println("异常"+e);
                   }
            }
      }
 }
}
class xiancheng3 extends Thread{
 public void run(){
      for(int i=0;i<20;i++)
      {
            System.out.print("☆");
            try
            {
                Thread.sleep(1000);
            }
            catch(InterruptedException e)
            {
                System.out.println("异常"+e);
```

```
            }
        }
    }
   }
   public class Deng{
    public static void main(String args[])
    {
        xiancheng mt1=new xiancheng();
        xiancheng2 mt2=new xiancheng2();
        xiancheng3 mt3=new xiancheng3();
         mt2.mt=mt1;
        mt1.start();
        mt2.start();
        mt3.start();
    }
   }
```

运行代码，得到如图 12-9 所示的结果。

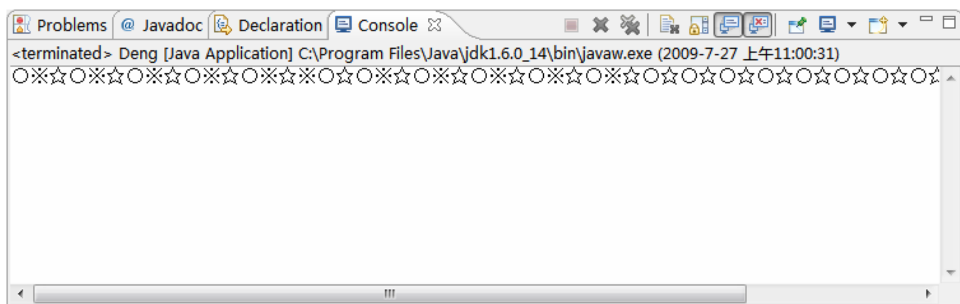
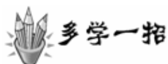


图 12-9　线程进行等待



在上面的一个实例中，讲解了线程中的 join（）方法，下面再展示一段代码，同样有 join() 方法让线程进入等待，其代码见“光盘：源代码/第 12 章/Dengone.java”：

```
   class Sleeper extends Thread
   {
    private int sleeptime;
    public Sleeper(String name,int sleepTime){
        super(name);
        sleeptime=sleepTime;
        start();
    }
    public void run(){
        try{
            for(int i=0;i<5;i++){
                System.out.println(getName()+"打印 "+i);
                sleep(sleeptime);
```

```
            }
        }catch(Exception e){
            System.out.println(getName()+" interrupted");
        }
        System.out.println(getName()+" 执行完毕");
    }
  }
  class Joiner extends Thread{
   private Sleeper sleeper;
   public Joiner(String name,Sleeper sleeper){
        super(name);
        this.sleeper=sleeper;
        start();
    }
   public void run(){
        try{
            System.out.println(getName()+" 开始执行");
            System.out.println(sleeper.getName()+"开始使用 join()方法");
            sleeper.join();
            for(int i=5;i>0;i--){
              System.out.println(getName()+"打印 "+i);
                           }
            }catch(Exception e){
                e.printStackTrace();
        }
        System.out.println(getName()+" 执行完毕");
    }
  }

  public class Dengone{
   public static void main(String[] args) {
   // TODO 自动生成方法存根
        Sleeper sleeper1=new Sleeper("线程 A sleeper1",1500),
                sleeper2=new Sleeper("线程 A sleeper2",1500);
        Joiner joiner=new Joiner("线程 B",sleeper2);
        sleeper1.interrupt();
    }
  }
```

运行代码，得到如图 12-10 所示的效果。

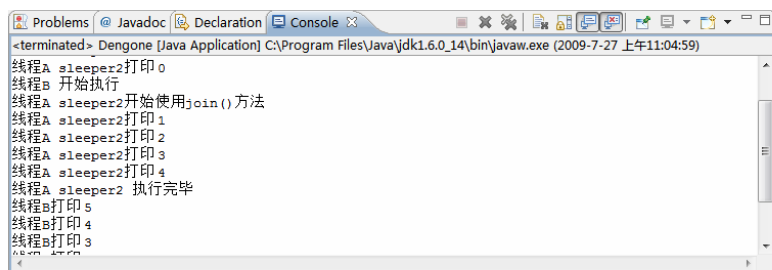


图 12-10　join 方法的特效

12.5.2 当前线程进入睡眠状态

在编写程序的时候，有时需要让当前线程停止一段时间，然后再继续执行。Java 语言提供了 sleep()方法，可以实现这种功能。sleep()方法称做睡眠方法，此方法的时间由毫秒决定。

实例 61：使用 sleep()方法让当前进程停止一段时间然后再执行

下面将给出一段程序，使用 sleep()方法让当前进程停止一段时间然后再执行，其代码见“光盘：源代码/第 12 章/Deng.java”：

```
public class Shuione{
 public static void main(String args[])
 {
     Shuione1 mt1=new Shuione1();
     Shuione2 mt2=new Shuione2();
     Shuione3 mt3=new Shuione3();
     mt1.start();
     mt2.start();
     mt3.start();
 }
}
class Shuione1 extends Thread{
 public void run()
 {
     for(int i=0;i<20;i++)
     {
         System.out.print("●");
         try
         {
             Thread.sleep(1000);
         }
         catch(InterruptedException e)
         {
             System.out.println("异常"+e);
         }
     }
 }
}
class Shuione2 extends Thread{
 public void run()
 {
     for(int i=0;i<20;i++)
     {
         System.out.print("■");
         try
         {
```

```
                Thread.sleep(1000);
            }
            catch(InterruptedException e)
            {
                System.out.println("异常"+e);
            }
        }
    }
}
class Shuione3 extends Thread{
 public void run()
 {
     for(int i=0;i<20;i++)
     {
         System.out.print("▲");
         try
         {
             Thread.sleep(1000);
         }
         catch(InterruptedException e)
         {
             System.out.println("异常"+e);
         }
     }
 }
}
```

运行代码，得到如图 12-11 所示的结果。

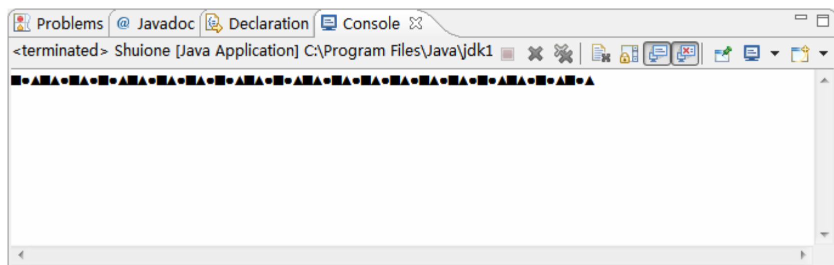


图 12-11　线程进入睡眠状态

多学一招

在上面的一个实例中，讲解了线程中的 sleep（）方法，下面再展示一段代码，使用 sleep()方法让当前进程停止一段时间然后再执行，其代码见“光盘：源代码/第 12 章/Dengtwo.java”：

```
public class Dengtwo extends Thread
{
```

```
        int count=0;
        public Dengtwo()
       {
            start();
        }
        public String toString()
       {
            return super.toString()+(count++);
        }
        public void run()
       {
            int count=3;
            while(true)
       {
                System.out.println(this);
                Try
       {
                    sleep(1000);
                }
       catch(Exception e){
                    System.out.println("i am interrupted");
                }
                count--;
                if(count==0)return;
              }


        }
        public static void main(String[] args){
                for(int i=0;i<3;i++){
                    new Dengtwo();
                }
                new Dengtwo();
                    }
       }
```

运行代码，得到如图 12-12 所示的结果。

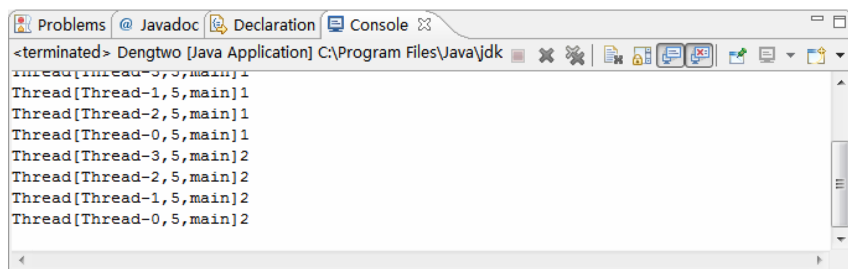


图 12-12　sleep()方法

12.5.3　当前线程做出让步

yield()方法是最具有品德的一种方法，它被称为让步方法，就好比两个人过独木桥有着同等的机会，如果都争抢着过，不拼个你死我活是很难解决问题的。而 yield()方法问世彻底解决了编程中的类似难题，让优先级相同的线程获得运行的机会，而自己这个线程做出让步，下面通过一段代码进行讲解，其代码见“光盘：源代码/第 12 章/Rang.java”：

```
public class Rang
{
    public static void main(String args[])
    {
        Rang1 mt1=new Rang1();
        Rang2 mt2=new Rang2();
        Rang3 mt3=new Rang3();
        mt1.start();
        mt2.start();
        mt3.start();
    }
 }

 class Rang1 extends Thread
 {
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            System.out.print("●");
            Thread.yield();
        }
    }
 }

 class Rang2 extends Thread
 {
    public void run()
    {
        for(int i=0;i<10;i++)
        {
            System.out.print("■");
            Thread.yield();
        }
    }
 }

 class Rang3 extends Thread
 {
```

```
        public void run()
    {
            for(int i=0;i<10;i++)
    {
                System.out.print("▲");
            }
        }
    }
```

运行代码，得到如图 12-13 所示的结果。

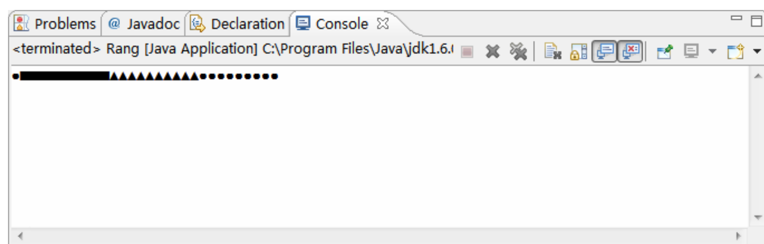


图 12-13　线程做出让步

12.6　多线程同步

当多个线程访问同一资源时，有必要限制这个资源一次只能让一个线程访问，这就是线程的同步。就像过独木桥，一次只能过一个人，如两个人同时过就会拥堵，这个时候就必须将其控制。为了避免这种情况发生，于是就有了多线程的同步。

12.6.1　同步的重要性

在 Java 程序中，多线程同步处理十分重要，如一个电子商务平台出售一种图书，甲、乙在同一时间各抢购 60 本，而库存只有 100 本，明显库存不够。要处理好类似的情况，就需要用到多线程同步的功能，下面通过一段代码讲解多线程同步的重要性，其代码见“光盘：源代码/第 12 章/Tong.java”：

```
public class Tong
{

 public static void main(String args[])
 {
     ShangDian1 sd=new ShangDian1(100);
     GouMai1 no1=new GouMai1(60,sd,"NO1");
     GouMai1 no2=new GouMai1(80,sd,"NO2");
     no1.start();
     no2.start();
 }
}
```

```
class ShangDian1
{
 int kucun=0;
 public ShangDian1(int kucun)
 {
     this.kucun=kucun;
 }

 public void goumai(int i)
 {

     if(i<kucun)
     {
         try
         {
             Thread.sleep(1000);
         }
         catch(InterruptedException e)
         {
             e.printStackTrace();
         }

         kucun-=i;
         System.out.println(Thread.currentThread().getName()+"购买"+i+"
《他的国》");
         System.out.println("书店剩余"+kucun+"《他的国》");
     }
     else
     {
         System.out.println("库存量不够");
     }
 }
}

class GouMai1 extends Thread
{

 int i;
 ShangDian1 sd;

 public GouMai1(int i,ShangDian1 sd,String name)
 {
     this.i=i;
     this.sd=sd;
     this.setName(name);
```

```
    }
    public void run()
    {
        sd.goumai(i);
    }
  }
```

运行代码，得到如图 12-14 所示的结果。

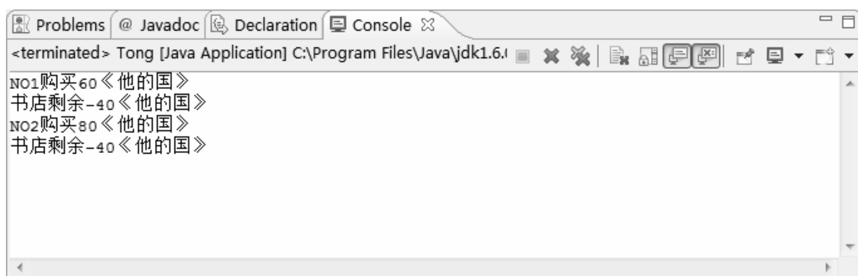


图 12-14　线程同步

多学一招

线程同步十分重要，如果遇到上面的问题而不同步，就会出现严重的库存不足，这对于购物网站来说相当重要。

12.6.2　轻松实现同步

在上面的章节中，用户已经初步体会到了同步的重要，那么如何才能实现同步呢？实现同步十分简单，访问资源用户可以用关键字 synchronzied 标记，这样在需要调用这个方法的线程执行完成之前，其他调用标志为 synchronized()的方法会被堵塞。读者可以使用下面的方法声明，其格式如下：

```
Synchronized void sun
{
//实现一个取和的同步方法
}
```

也可以定义一个取最大值的方法，其格式如下：

```
Synchronized void max
{
//实现一个取最大值的同步方法
}
```

实例 62：使用线程同步让程序符合现实要求

下面将给出一段程序，使用线程同步让程序符合现实要求，其代码见“光盘：源代码/第 12 章/Tongone.java”：

```
public class Tongone extends Thread
{
 private String cha;
 static Object printer=new Object();
 public Tongone(String cha)
{
     this.cha=cha;
 }
 void print()
{
     synchronized(printer)
{
     for(int i=1;i<=5;i++)
         System.out.print(cha+" ");
     }
 }
 public void run()
{

         for(int i=1;i<5;i++)
{
             print();
             System.out.println();

     }
 }
 public static void main(String[] args)
{
     // TODO 自动生成方法存根
     Tongone test1=new Tongone("甲线程");
     Tongone test2=new Tongone("乙线程");
     Tongone test3=new Tongone("丙线程");
     test1.start();
     test2.start();
     test3.start();
         }
}
```

运行代码，得到如图 12-15 所示的结果。

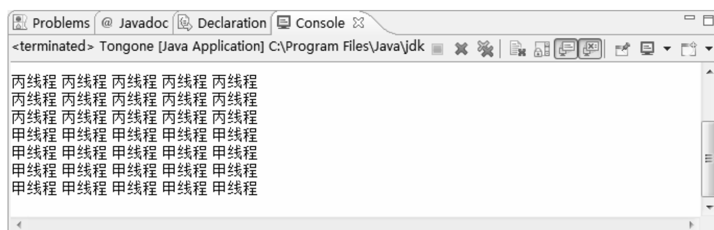
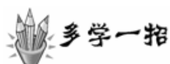


图 12-15　实现同步



上面实例里讲解了如何实现同步，下面将对代码进行修改，以实现同步。修改后的代码见“光盘：源代码/第 12 章/Tongtwo.java”：

```
public class Tongtwo
{
 public static void main(String args[]) throws Exception
 {
     ShangDian0 sd=new ShangDian0(100);
     GouMai0 no1=new GouMai0(70,sd,"NO1");
     GouMai0 no2=new GouMai0(80,sd,"NO2");
     no1.start();
     no2.start();
 }
}
class ShangDian0
{
 int kucun=0;
 public ShangDian0(int kucun)
 {
     this.kucun=kucun;
 }
 public synchronized void goumai(int i)
 {

     if(i<kucun)
     {
         kucun-=i;
System.out.println(Thread.currentThread().getName()+"购买"+i+"本《他的国》");
         System.out.println("商店剩余"+kucun+"本《他的国》");
     }
     else
     {
         System.out.println("库存量不够");
     }
 }
}
class GouMai0 extends Thread
{
 int i;
 ShangDian0 sd;
 public GouMai0(int i,ShangDian0 sd,String name)
 {
     this.i=i;
```

```
        this.sd=sd;
        this.setName(name);
    }
    public void run()
    {
        sd.goumai(i);
    }
}
```

运行代码，得到如图 12-16 所示的结果。

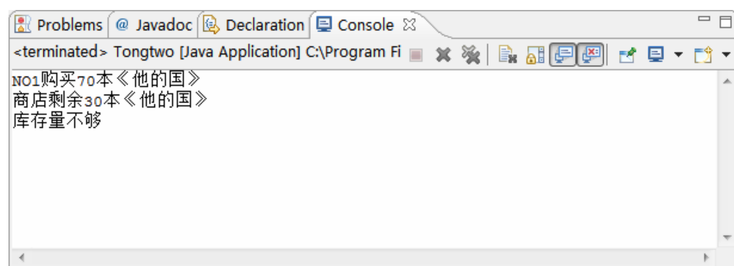


图 12-16　实现同步

12.6.3　什么是死锁

死锁是每一个程序员都唯恐避之不及的东西，有许多人将它比做黑白无常、幽灵或者魔鬼。其实程序员遇见死锁的概率很低，或许一辈子都不会遇见，但是一旦遇到死锁，就会导致程序崩溃。在一个大程序中，死锁是很难被发现的。对于一个程序员来说，一定要注意死锁的问题。下面通过一段代码讲解死锁，其代码见“光盘：源代码/第 12 章/Sisuo.java”：

```
public class Sisuo
{
    public static void main(String args[])
    {
        String s1="s1";
        String s2="s2";
        MyThread0 mt1=new MyThread0(s1,s2,"NO1");
        MyThread0 mt2=new MyThread0(s2,s1,"NO2");
        mt1.start();
        mt2.start();
    }
}
class MyThread0 extends Thread
{
    String s1,s2;
    public MyThread0(String s1,String s2,String name)
    {
        this.s1=s1;
```

```
        this.s2=s2;
        this.setName(name);
    }
    public void run()
    {
        synchronized(s1)
        {
            System.out.println(this.getName()+"锁住"+s1);
            try
            {
                Thread.sleep(10);
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            synchronized(s2)
            {
                System.out.println(this.getName()+"锁住"+s2);
                System.out.println("线程结束");
            }
        }
    }
}
```

运行代码，得到如图 12-17 所示的结果。

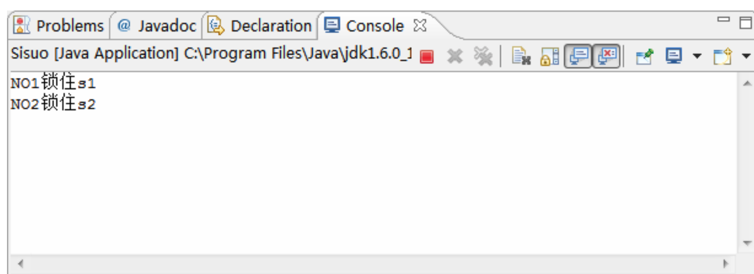


图 12-17　死锁

通过上面的程序不难理解线程是可以阻塞的，并且其具有的同步控制机制可以防止其他线程在锁没有释放的情况下访问这个对象，这时就产生了矛盾、当出现线程 A 在等待线程 B，而线程 B 又在等待线程 A 的情况时，就会造成死锁。那么如何判断死锁呢？方法十分简单，死锁满足下面的所有要求：

- **互斥条件**：线程使用的资源必须至少满足有一个不能共享的。
- **请求与保持条件**：至少一个线程必须持有一个资源并且正在等待获取一个当前被其他线程持有的资源。
- **非剥夺条件**：分配的资源不能从相应的线程中被强制剥夺。
- **循环条件**：第一个线程等待其他线程，后者又在等待第一个线程。

12.7　线程之间互相通信

不同的线程完成不同的任务，但在执行任务的时候线程之间会有一定的联系，这就需要使这些线程互相通信。在 Java 程序中，用户可以使用以下三种方法进行线程间的通信，现介绍如下：

- Wait()：让当前线程进入等待状态，直到其他线程调用 notify()方法。
- Notify()：恢复一个等待状态中的线程。
- NotifyAll()：恢复所有等待状态中的线程。

三个方法的定义格式如下：

- Final void wait() throws InterruptedException。
- Final void notify()。
- Final void notifyAll()。

其中，Wait()方法存在重载，可以添加参数，表示时间。

实例 63：在线程之间互相通信

下面将给出一段程序，使线程之间互相通信，其代码见“光盘：源代码/第 12 章/Tongxin.java”：

```
class Water
{
 //水塘类
 static Object water=new Object();
 static int total=6;
 //假设水塘总共可以含水量为 6
 static int mqsl=3;
 //假设水塘中拥有含水量为 3
 static int ps=0;
 //假设水塘目前排水量为 0
}

class ThreadA extends Thread
{
//排水类
 void pswork()
 {
     synchronized(Water.water)
     {
         System.out.println("水塘是否没有水： "+isEmpty());
         if(isEmpty())
         {
             try
             {
                 Water.water.wait();
             }catch(InterruptedException e)
             {
```

```
                    e.printStackTrace();
                }
            }
            else
            {
                Water.ps++;
                System.out.println("水塘目前排水水量 "+Water.ps);
            }
                    }
  }

  public void run()
  {
      while(Water.mqsl<Water.total)
 {
            if(isEmpty())
                System.out.println("水塘目前没有水,排水线程被挂起");
            System.out.println("排水工作开始");
            pswork();
            try
 {
                sleep(1000);
            }catch(InterruptedException e)
 {
                e.printStackTrace();
            }

      }
  }
  public boolean isEmpty()
 {
      return Water.mqsl==Water.ps?true:false;
  }
 }

 class ThreadB extends Thread
 {
 //进水类
  void jswork()
 {
      synchronized(Water.water)
 {
            Water.mqsl++;
 //假设水塘每小时进水量为 1
            Water.water.notify();
            System.out.println("水塘目前进水量为 "+Water.mqsl);
      }
  }
```

```
    public void run()
    {
        while(Water.mqsl<Water.total)
    {
            System.out.println("进水工作开始");
            jswork();
            try{
                sleep(3000);
            }catch(InterruptedException e){
                e.printStackTrace();
            }
        }
    }
    }

    public class Tongxin
    {

    public static void main(String[] args)
     {
        ThreadA threadA=new ThreadA();
        ThreadB threadB=new ThreadB();

        threadB.start();
        threadA.start();
     }
    }
```

运行代码，得到如图 12-18 所示的结果。

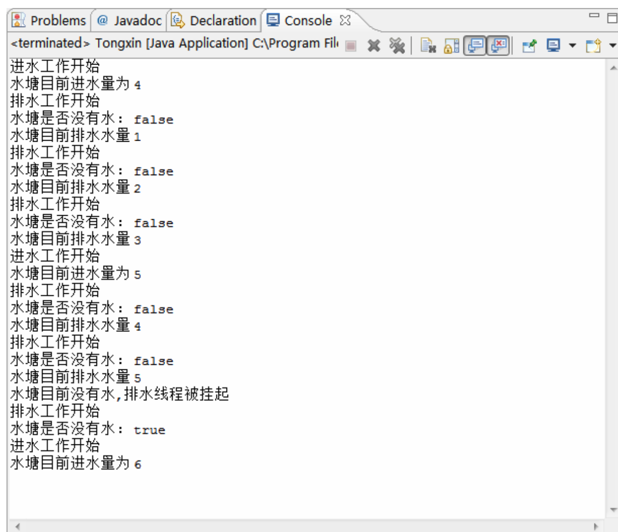


图 12-18　线程之间的通信

12.8　疑难问题解析

本章详细介绍了线程初步、创建单线程、创建多线程等的基本知识。本节将对本章中较难理解的问题进行深入讲解。

读者疑问：本章讲解了线程、多线程，又简单地讲解了进程，请问线程和进程之间有什么区别和联系呢？

解答：线程和进程有着十分紧密的联系，一个进程里有多个线程，当然，也允许一个进程里只有一个线程，有线程必然有进程。

读者疑问：前面讲解了死锁，死锁是程序员最害怕的东西，那么如何才能避免一个程序出现死锁呢？

解答：初学者不必担心死锁的问题，一个程序员遇见死锁的概率比遇见鬼还低，如果真遇见了，也只能按照前面讲解的原则仔细检查程序的状态，仔细排除。



职场点拨——揣测老板的弦外之音

在很多时候，上级为了解真实情况或职员的态度，会“挑逗”性地说一些话进行试探，此时我们一定要谨慎言行。

假如一天领导找你单独谈话，谈着谈着就露出诚恳又凝重的神色，说“这件事我只对你一个人说……”。这时候你就不可对这句话等闲视之，要相当重视他话里的弦外之音。一般来说，这句话的弦外之音有如下几种情况。

1）提高你谈话的兴趣，倘若是这种情况，大可不必太当真。

2）领导可能对你同事的工作不满意，却无法开口，他只好向你发发牢骚。此时你不需要在意，不要和领导斗气，建议一笑了之。

3）领导说这话是因为怕你把事情传出去，倘若传出去，将会问责于你，此时你要守口如瓶。

总之，领导也是人，只不过他比我们的社会经验更丰富。在日常生活和工作中，我们要善于分析领导们说的话，只有这样，才能在职场里混得如鱼得水。

温故而知新——第二篇实战范例

第二篇学习的重点就是面向对象，下面来回顾本篇所学的知识，使读者对这些知识的理解得到升华。

范例 1　类的继承

在编写程序的时候，类的继承是十分常见的，正因为如此，它才能体现面向对象编程的特点。下面展示一段代码，以巩固初学者对类的继承的深层次理解，其代码见“光盘：源代码/温故而知新 2/Jicheng.java”：

```
public class Jicheng
{
 String bname;
 int    bid;
 int    bprice;
     Jicheng()
 {
     bname="雀巢咖啡";
     bid=8008691;
     bprice=18;
 }

 Jicheng(Jicheng a)
 {
     bname=a.bname;
     bid=a.bid;
     bprice=a.bprice;
 }

 Jicheng(String name,int id,int price)
 {
     bname=name;
     bid=id;
     bprice=price;

 }

 void print()
```

```
  {
    System.out.println("产品名："+bname+"  产品号："+bid+"  价格："+bprice);
  }
}

class Badder extends Jicheng
{
 String badder;

 Badder()
 {
     super();
     badder="广东省东莞市";
 }

 Badder( Badder b)
 {
     super(b);
     badder=b.badder;
 }

 Badder(String x,int y,int z,String aa)
 {
     super(x,y,z);
     badder=aa;
 }
}

class Factory extends Badder
{
 String factory;

 Factory()
 {
     super();
     factory="广东省东莞市";
 }

 Factory(Factory c)
 {
     super(c);
     factory=c.factory;
 }

 Factory(String x,int y,int z,String l,String n)
 {
```

```
        super(x,y,z,l);
        factory=n;
    }
   }

   class textone
   {
    public static void main(String args[])
    {
        Factory a1=new Factory();
        Factory a2=new Factory("咖啡伴侣",8008707,25,"广东省东莞市","广东");
        Factory a3=new Factory(a2);
        System.out.println(a1.badder);
        System.out.println(a1.factory);
        a1.print();
        System.out.println(a2.badder);
        System.out.println(a2.factory);
        a2.print();
        a3.print();
    }
   }
```

运行代码，得到如范例图 2-1 所示的结果。



范例图 2-1　类的继承

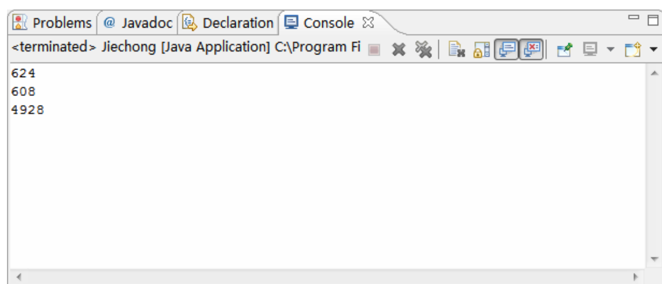
范例 2　接口的实现

在面向对象的程序中，除了类的继承是面向对象的特性，接口的实现也是面向对象的主要特性之一。下面将讲解一段代码，以巩固初学者对接口的理解，其代码见“光盘：源代码/温故而知新 2/Jiechong.java”：

```
   interface Aone
   {
    void add(int a,int b);
   }
```

```
interface Atwo
{
 void sub(int a,int b);
}
//创建 Athree 接口继承了 Aone，Atwo 接口
interface Athree extends Aone,Atwo
{
 void mul(int a,int b);
}
class JieXian implements Athree
{
 public void add(int a,int b)
 {
     System.out.println(a+b);
 }
 public void sub(int a,int b)
 {
     System.out.println(a-b);
 }
     public void mul(int a,int b)
 {
     System.out.println(a*b);
 }
}
class Jiechong
{
 public static void main(String args[])
 {
     JieXian aa=new JieXian();
     aa.add(616,8);
     aa.sub(616,8);
     aa.mul(616,8);
 }
}
```

运行代码，得到如范例图 2-2 所示的结果。



范例图 2-2　执行结果

范例 3　异常的处理

在 Java 程序设计中，异常是无法避免的。在任何一个工程中，都难免需要处理异常，否则这个程序就是不完整的。下面将通过一段代码，让初学者对异常的处理有着更深的理解和认识，其代码见“光盘：源代码/温故而知新 2/Chang.java”：

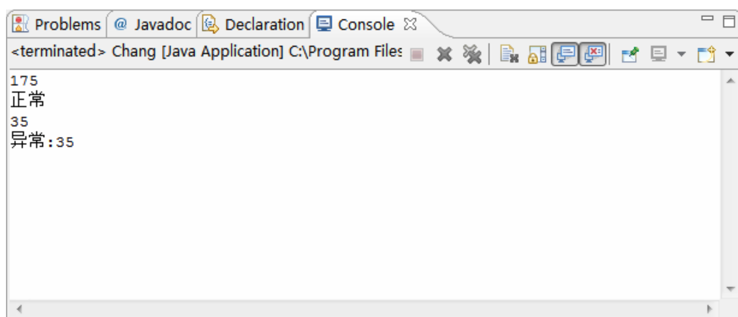
```
class MyException extends Exception
{
 private int chengji;
 MyException(int a)
 {
    chengji=a;
 }
 public String toString()
 {
    return "异常:"+chengji;
 }
}

public class Chang
{
 void methodName(int a) throws MyException
 {
    System.out.println(a);
    if(a<60)
    {
        throw new MyException(a);
    }
    else
    {
        System.out.println("正常");
    }
 }

 public static void main(String args[])
 {
    Chang yc=new Chang();
    try
    {
        yc.methodName(175);
        yc.methodName(35);
    }
    catch(MyException e)
    {
        System.out.println(e);
    }
```

```
    }
  }
```

运行代码，得到如范例图 2-3 所示的结果。



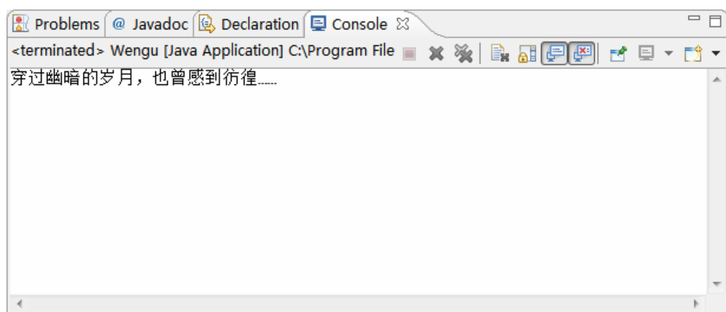
范例图 2-3　异常处理

范例 4　读取文件的字符

在 Java 程序中，用户可以通过程序操作文件里的字符，这种操作就是文件的 I/O 处理。下面通过一段代码回顾文字的处理，其代码见“光盘：源代码/温故而知新 2/Wengu.java”：

```
import java.io.*;
public class Wengu
{
 public static void main(String args[])
 {
     try
     {
         FileReader fr=new FileReader("wengu.txt");
         BufferedReader br=new BufferedReader(fr);
         String a;
         while((a=br.readLine())!=null)
         {
             System.out.println(a);
         }
         fr.close();
         br.close();
     }
     catch(IOException e)
     {
         System.out.println(e);
     }
         }
}
```

在执行程序前，必须要将 wengu.txt 放在一定的位置，如果是使用 eclipse 开发项目，用户可将这个而文件放在项目的根目录下。运行代码，得到如范例图 2-4 所示的程序结果，



范例图 2-4　读取文件的字符

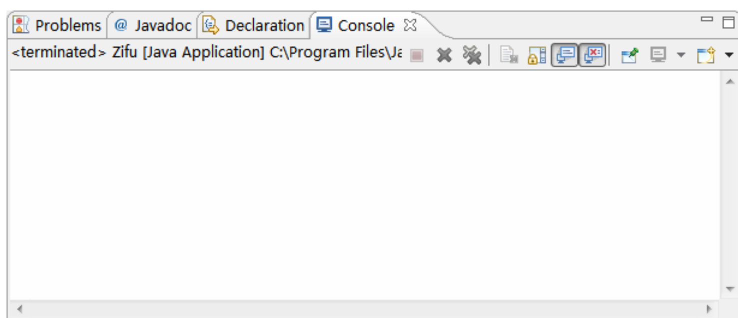
范例 5　缓冲字节流

缓冲字节流只能处理英文和数字字符，而无法处理汉字字符，当然用户也可以使用缓冲字符流处理英文和数字字符，但是速度会比缓冲字节流慢。下面通过一段代码讲解缓冲字节流，其代码见“光盘：源代码/温故而知新 2/Zifu.java”：

```
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class Zifu
{
 public static void main(String args[]) throws IOException
 {
     FileInputStream bu1;
     FileOutputStream bu2;
     BufferedInputStream bu3;
     BufferedOutputStream bu4;
     int i;
     try
     {
         bu1=new FileInputStream("ming1.txt");
         bu3=new BufferedInputStream(bu1);
         bu2=new FileOutputStream("ming2.txt");
         bu4=new BufferedOutputStream(bu2);
                  i=bu3.read();
         while(i!=-1)
         {
          bu4.write(i);
          bu4.flush();
          i=bu3.read();
```

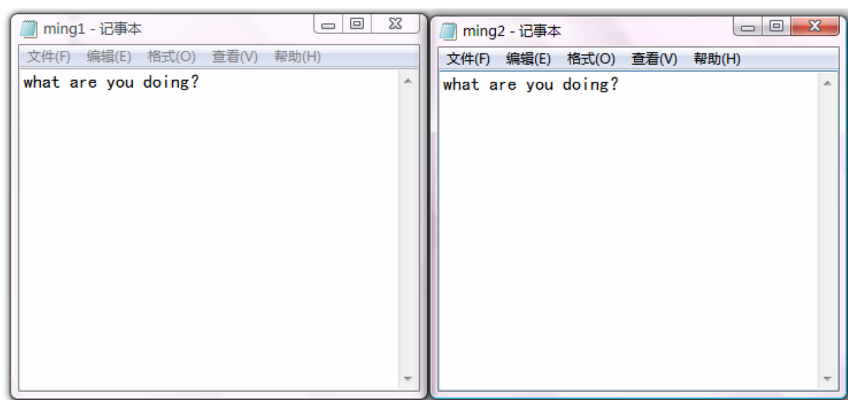
```
                }
                bu1.close();
                 bu2.close();
                 bu3.close();
                 bu4.close();
              }
            catch(IOException e)
              {
                    System.out.println("没有找到文件");
              }
        }
      }
```

将文件 ming1.txt 和 ming2.txt 放在一定的位置，运行代码，得到如范例图 2-5 所示的结果。



范例图 2-5　执行结果

打开两个文件，进行复制，得到如范例图 2-6 所示的结果。



范例图 2-6　执行结果

范例 6　深刻认识多线程

线程是程序中必不可少的，对单线程可以不做处理，但是对于多线程则需要进行一些必

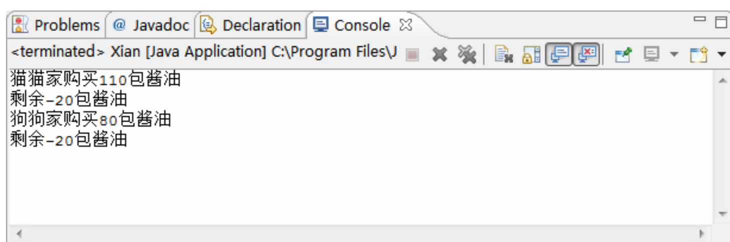
要的操作。下面通过一段代码复习对线程进行处理，其代码见“光盘：源代码/温故而知新2/xian.java”：

```
public class Xian
{
 public static void main(String args[])
 {
    ShangDian1 sd=new ShangDian1(170);
    GouMai1 no1=new GouMai1(110,sd,"猫猫家");
    GouMai1 no2=new GouMai1(80,sd,"狗狗家");
    no1.start();
    no2.start();
 }
}
class ShangDian1
{
 int kucun=0;
 public ShangDian1(int kucun)
 {
    this.kucun=kucun;
 }
 public void goumai(int i)
 {

    if(i<kucun)
    {
        try
        {
            Thread.sleep(1000);
        }
        catch(InterruptedException e)
        {
            e.printStackTrace();
        }
        kucun-=i;
        System.out.println(Thread.currentThread().getName()+"购买"+i+"
包酱油");
        System.out.println("剩余"+kucun+"包酱油");
    }
    else
    {
        System.out.println("库存量不够");
    }
 }
}
class GouMai1 extends Thread
{
```

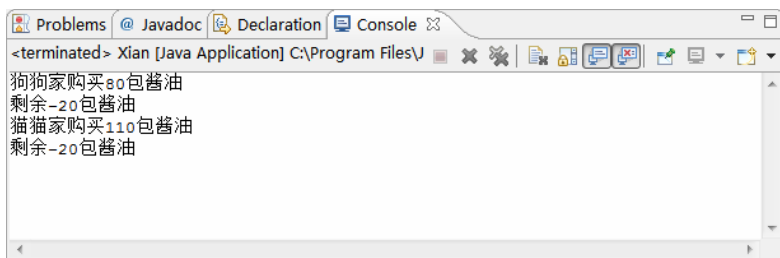
```
        int i;
        ShangDian1 sd;
        public GouMai1(int i,ShangDian1 sd,String name)
        {
            this.i=i;
            this.sd=sd;
            this.setName(name);
        }
        public void run() {
            sd.goumai(i);
        }
      }
```

运行代码，得到如范例图 2-7 所示的结果。



范例图 2-7　认识多线程

再次运行代码，结果可能会发生变化，如范例图 2-8 所示。



范例图 2-8　再次执行的结果

提示

试想一下，为什么程序执行时结果会不同？

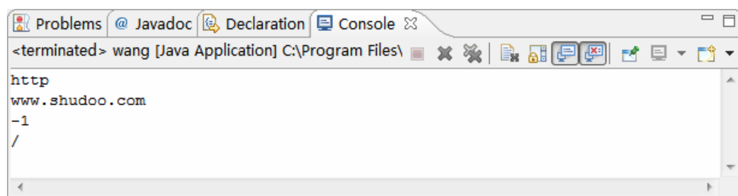
范例 7　认识网络编程

URL 是网络编程最常用的协议之一，它有许多对象和属性，如它的协议名、主机名、端口号等。对象产生后，其属性是不能改变的，在 URL 的 API 中定义了很多方法来获得这些属性。下面通过一段代码进行预习，其代码见“光盘：源代码/温故而知新 2/wang.java”：

```
import java.io.*;
import java.net.*;
```

```
public class wang
{
 public static void main(String args[])
 {
     try
     {
     URL u1=new URL("http://www.shudoo.com/");
//获取该 URL 的协议名
     System.out.println(u1.getProtocol());
//获取该 URL 的主机名
     System.out.println(u1.getHost());
//获取该 URL 的端口号
     System.out.println(u1.getPort());
//获取该 URL 的文件名
     System.out.println(u1.getFile());
    }
    catch(MalformedURLException e)
    {
     System.out.println(e);
    }
     catch(IOException ee)
    {
     System.out.println(ee);
    }
     catch(Exception eee)
    {
     System.out.println(eee);
    }
 }
}
```

运行代码，得到如范例图 2-9 所示的结果。



范例图 2-9　网络编程

第三篇　提　高　篇

第 13 章　网络与通信

互联网改变了人们的生活方式，现如今几乎所有程序都离不开网络，人们早已习惯使用网络快速传递信息，比如银行系统、ATM 机器等等。Java 在网络通信方面的开发特点突出，其技术之成熟要远远领先于其他语言。本章将讲解 Java 在网络通信方面的开发。本章主要内容如下：

- 什么是网络通信。
- 网络编程的初步。
- 职场点拨——同事相处之道。

2010 年 XX 月 XX 日，多云

不知不觉间工作半年了，大部分时间都窝在办公室里，面对着同事。我愈发感觉把握好同事关系很重要。虽然有时其乐融融，有时也会为了某一个观点而辩论甚至争吵，但是我感觉我在进步，明白了更多的处事之道。



一问一答

小菜：“和同事相处是一门学问，我一定要和他们好好相处。”

Wisdom：“呵呵，只要你在职场中，就需要面对同事关系。同事能在技术上给你点拨，并且能助你成长。即使在激烈的辩论中，也能使你意识到问题的所在，提高你的业务能力。所以你要好好处理同事关系，和同事在一起能学习到不少的生活知识。”

小菜：“恩，同事关系很重要。言归正传，网络通信开发需要掌握什么知识？”

Wisdom：“网络通信开发肯定要遵循一些协议，不然难以传播，如学计算机的基础所提出的 HTTP 协议、TCP 协议等等。”

13.1　什么是网络通信

什么是网络通信？在学习具体知识之前，用户一定要熟悉网络方面的专业术语，只有弄

懂这些知识点，才能更好地掌握网络编程。

13.1.1　TCP/IP 协议

TCP/IP（Transmission Control Protocol/Internet Protocol）的简写，中文译名为传输控制协议/互联网络协议，是Internet最基本的协议，简单地说，就是由底层的 IP 协议和 TCP 协议组成的。众所周知，如今计算机上因特网都要进行 TCP/IP 协议设置，显然该协议已经成了当今地球村“人与人”之间的“牵手协议”。

IP（Internet Protocol）协议的英文名直译就是因特网协议，从名称就不难看出 IP 协议的重要性。在现实生活中，人们都是把货物包装成一个个的纸箱或者是集装箱之后再进行运输，在网络世界中各种信息也是通过类似的方式进行传输的。IP 协议规定了数据传输时的基本单元和格式。如果比作货物运输，那么 IP 协议即是规定了货物打包时的包装箱尺寸和包装的程序。除了这些以外，IP 协议还定义了数据包的递交办法和路由选择。在货物运输中，IP 协议规定规定的就是货物的运输方法和运输路线。

IP 协议已经规定了数据传输的主要内容，那 TCP（Transmission Control Protocol）协议是做什么的呢？不知大家发现没有，在 IP 协议中定义的传输是单向的，也就是说发出去的货物对方是否收到发货方是不知道的，就好像 8 毛钱一份的平信一样。那对于重要的需要寄挂号信的信件该怎么办呢？TCP 协议就可实现寄“挂号信”。TCP 协议提供了可靠的面向对象的数据流传输服务的规则和约定。在 TCP 模式中，假如对方发一个数据包过来，本方也要发一个确认数据包回去，通过这种方式来增加传输的可靠性。

13.1.2　使用 URL 进行网络链接

URL 是 Uniform Resource Locator 的缩写，称为网页地址。是因特网上标准的资源地址。它最初是由蒂姆·伯纳斯 • 李发明的用来作为万维网地址的，如今它已经被万维网联盟编制为因特网标准 RFC1738 了，用于完整地描述 Internet 上网页和其他资源的地址。

Internet 上的每一个网页都具有一个标识，通常称之为 URL 地址，这种地址可以是本地磁盘，也可以是局域网中的某一台计算机，更多的是 Internet 上的站点。简单地说，URL 就是 Web 地址，俗称“网址”。

在 Java 中，有一个 URL 类，它在 Java.net 包中，是网络编程的重要内容。它为 Java 访问网络资源提供了接口，通过这些接口可以很容易地浏览服务器上的文件，常用的构造方法的格式如下：

- public URL(String spec)：通过一个 URL 地址的字符串构造一个 URL 对象，如 URL a=new URL(“http://www.baidu.com”)。
- public URL(URL context，String spec)：通过 URL 和相对 URL 指定文件构造一个 URL 对象，如 URL b=new URL(“index.jsp”)。
- public URL(String protocol,String host,String file)：通过协议名、主机名和相对的 URL 指定文件构造一个 URL 对象，如 URL c= new URL(“http”, “www.taobao.com.cn”, “download/index.html”)。
- public URL(String protocol,String host,int port,String file)：通过协议名、主机名和端口号和相对的 URL 指定文件构造一个 URL 对象，如 URL d= new URL(“http”,

“www.qq.com”, “6870”,“download/index.html”)。

在构造 URL 时，可能会出现 MalformedURLException 异常，这时就需要处理该异常。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 13 章/URLone.java”：

```
import java.io.*;
import java.net.*;
public class URLone
{
 public static void main(String args[])
 {
     try
     {
        URL u1=new URL("http://www.taobao.com/");
     }
     catch(MalformedURLException e)
     {
        System.out.println(e);
     }
     catch(IOException ee)
     {
        System.out.println(ee);
     }
     catch(Exception eee)
     {
        System.out.println(eee);
     }
 }
}
```

13.1.3　编写 URL 程序常用的方法

在上面的那段程序，没有任何执行结果。正如之前所说的，URL 对象有很多属性，如它的协议名，主机名、端口号等。在对象产生后，这些属性是不能改变的。在 URL 的 API 中定义了很多方法来获得这些属性，下面列出了一些经常用到的方法：

- public String getProtocol()：获取该 URL 的协议名。
- public String getHost()：获取该 URL 的主机名。
- public int getPort()：获取该 URL 的端口号。
- public String getFile ()：获取该 URL 的文件名。
- public String getRef()：获取该 URL 在文件中的相对位置。
- public String getQuery()：获取该 URL 的路径。
- public String getPath()：获取该 URL 的路径。
- public String getAuthority()：获取该 URL 的权限信息。
- public String getUserInfo()：获得该 URL 的锚。

实例 64：使用 URL 的常用方法

下面将给出一段程序，讲解 URL 常用方法，其代码见“光盘：源代码/第 13 章/URL1.java”：

```
import java.io.*;
import java.net.*;
public class URL1
{
 public static void main(String args[])
 {
     try
     {
         URL u1=new URL("http://www.qq.com/");
     //获取该 URL 的协议名
         System.out.println(u1.getProtocol());
     //获取该 URL 的主机名
         System.out.println(u1.getHost());
     //获取该 URL 的端口号
         System.out.println(u1.getPort());
     //获取该 URL 的文件名
         System.out.println(u1.getFile());
     }
     catch(MalformedURLException e)
     {
         System.out.println(e);
     }
     catch(IOException ee)
     {
         System.out.println(ee);
     }
     catch(Exception eee)
     {
         System.out.println(eee);
     }
 }
}
```

运行代码，得到如图 13-1 所示的结果。

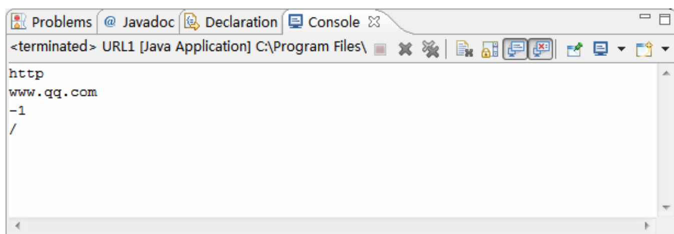
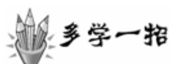


图 13-1　使用 URL



在上面的一个实例中，讲解了获取 URL 的一些常用信息，下面再展示一段代码，将前面讲过的 I/0 和 URL 的知识点结合进行讲解，其代码见“光盘：源代码/第 13 章/URLT.java”：

```
import java.io.*;
import java.net.*;
public class URLT
{
 public static void main(String args[])
 {
     try
     {
         //创建一个 URL 对象
         URL u1=new URL("http://www.hao123.com/");
         //构造一个 BufferedReader 对象
         BufferedReader br=new BufferedReader(
                    new InputStreamReader(u1.openStream()));
         String s;
         //从输入流不停的读数据，直到读完为止
         while((s=br.readLine())!=null)
         {
             //把读入的数据打印出来
             System.out.println(s);
         }
         //关闭输入流
         br.close();
     }
     catch(MalformedURLException e)
     {
         System.out.println(e);
     }
     catch(IOException ee)
     {
         System.out.println(ee);
     }
     catch(Exception eee)
     {
         System.out.println(eee);
     }
 }}
```

运行代码，得到如图 13-2 所示的结果。

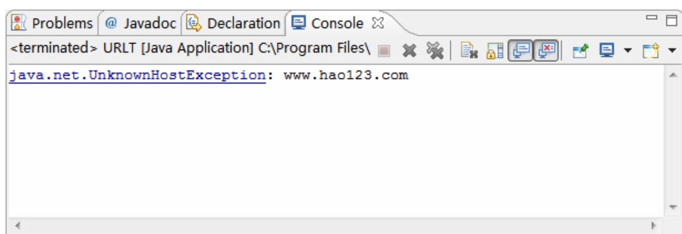


图 13-2　执行结果

提示：运行结果中有一个异常，产生这个异常是因为没有连接网络。

13.2　网络编程初步

网络编程可以分为创建 Socket、打开链接 Socket 的输入流和输出流入对 Socket 进行编程以及关闭 Socket 四步，在本节中将详细讲解。

13.2.1　创建 Socket

在 Java 的编写中，定义 Socket 和 ServerSocket 两个类，Socket 类表示客户端或者服务器 Socket 连接的两端。Socket 类下的构造方法如下：

- ❑ Socket(inetAddreess address,int port)。
- ❑ Socket(inetAddreess address,int port,boolean steam)。
- ❑ Socket(String host,int port)。
- ❑ Socket(String host,int port,boolean stream)。
- ❑ Socket(SocketImp imp1)。
- ❑ Socket(String host, int port ,inetAddreess localAddr,int localport)。
- ❑ Socket(inetAddreess address,int port, inetAddreess localAddr,int localport)。

参数介绍：

- ❑ address 表示 IP 地址。
- ❑ host 表示主机名。
- ❑ port 表示端口号。
- ❑ stream 指明 socket 数据类型为流或数据报。
- ❑ localPort 表示本地主机的端口号。
- ❑ localAddr 是本地地址。
- ❑ impl 是 socket 的父类。

在 Socket 类下有三个构造方法，其介绍如下：

- ❑ ServerSocket(int port)。
- ❑ ServerSocket(int port,int backlog)。
- ❑ ServerSocket(int port,int backlog,InetAddress binAddr)。

其中 bindAddr 表示本机地址

实例 65：使用 Socket

下面将给出两段代码讲解 Socket，一个是服务器端程序，一个是客户端程序。客户端程

序见“光盘：源代码/第 13 章/Kehu.java”：

```
import java.net.*;
import java.io.*;
public class Kehu
{
 public static void main(String args[])
 {
    try
    {
       Socket s=new Socket("127.0.0.1",2007);
       InputStream is=s.getInputStream();
       OutputStream os=s.getOutputStream();
       PrintStream ps=new PrintStream(os);
       ps.println("hello");
       DataInputStream dis=new DataInputStream(is);
       String request=dis.readLine();
       System.out.println(request);
       s.close();
    }
    catch(ConnectException e)
    {
       System.out.println("异常"+e);
    }
    catch(IOException ee)
    {
       System.out.println("异常"+ee);
    }
    catch(Exception eee)
    {
       System.out.println("异常"+eee);
    }
 }
}
```

服务端程序见“光盘：源代码/第 13 章/Fuwu.java”：

```
import java.net.*;
import java.io.*;
public class Fuwu
{
 public static void main(String args[])
 {
    try
    {
       ServerSocket ss=new ServerSocket(2007);
       while(true)
```

```
            {
                Socket s=ss.accept();
                InputStream is=s.getInputStream();
                OutputStream os=s.getOutputStream();
                DataInputStream dis=new DataInputStream(is);
                String request=dis.readLine();
                System.out.println(request);
                PrintStream ps=new PrintStream(os);
                ps.println("再见");
                s.close();
                ss.close();
            }
        }
        catch(IOException e)
        {
            System.out.println("异常"+e);
        }
        catch(Exception ee)
        {
            System.out.println("异常"+ee);
        }
    }
}
```

运行第一段代码，得到如图 13-3 所示的结果。

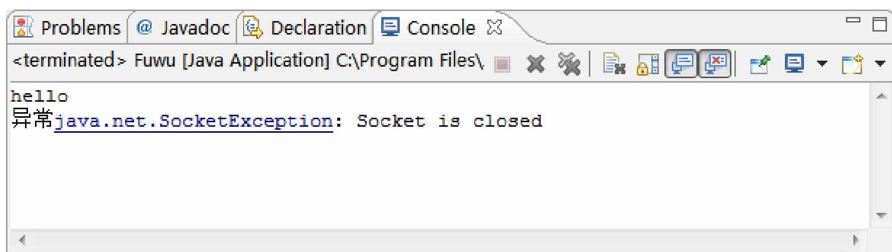


图 13-3　执行结果

运行第二段代码，得到如图 13-4 所示的结果。

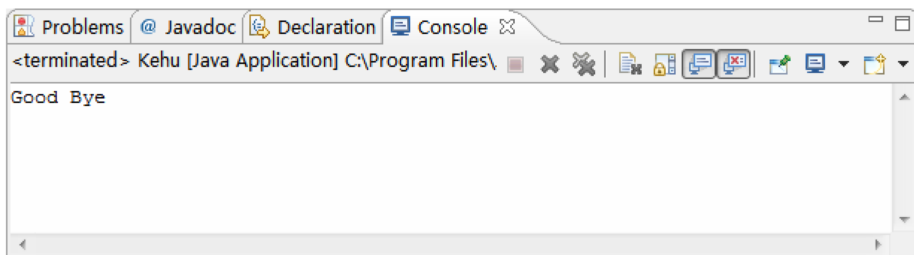


图 13-4　执行结果

13.2.2　多个客户端连接

在生活中，一个服务器程序肯定是由多个客户端进行连接的，下面展示一段代码，其代码见“光盘：源代码/第 13 章/Duoke.java”：

```
 import java.net.*;
import java.io.*;
public class Duoke
{
 public static void main(String args[])
 {
     boolean connected=true;
     try
     {
         ServerSocket ss=new ServerSocket(2007);
         int clientnum=0;
         while(connected)
         {
             ServerThread st=new ServerThread(ss.accept(),clientnum);
             st.start();
             clientnum++;
             System.out.println(clientnum);
         }
     }
     catch(Exception e)
     {
         System.out.println("异常"+e);
     }
 }
}

class ServerThread extends Thread
{
 private Socket s;
 int clientnum;
 public ServerThread(Socket s,int num)
 {
     this.s=s;
     clientnum=num+1;
 }

 public void run()
 {
     try
     {
         InputStream is=s.getInputStream();
```

```
            OutputStream os=s.getOutputStream();
            DataInputStream dis=new DataInputStream(is);
            String request=dis.readLine();
            System.out.println(request);
            PrintStream ps=new PrintStream(os);
            ps.println("Bye");
            s.close();
        }
        catch(Exception e)
        {
            System.out.println("异常"+e);
        }
    }
  }
```

运行代码，然后运行上一节讲解时用到的客户端程序的代码，调用多次，可得到如图 13-5 所示的结果。

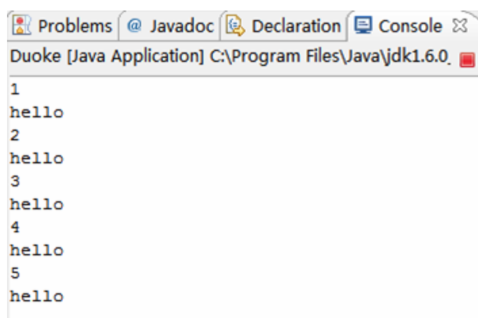


图 13-5　多个客户端连接

13.3　疑难问题解析

本章详细介绍了网络通信、网络编程的初步的基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：在本章中学习了通过 Java 编程来实现一些网络协议的方法，除了上面的协议外，还有其他的网络协议吗？那么 Java 程序是怎样实现的呢？

解答：在 Java 网络编程中，TCP/IP 和 URL 这两种协议使用十分频繁，也最为复杂，其他网络协议都可以以它们为模板进行改动后实现。

读者疑问：在 Java 网络编程中，对 socket 进行操作大致可分为多少步？

解答：网络编程首先是创建 socket、打开连接到 socket 的输入流和输出流，然后是对 socket 进行读写操作，最后是关闭 socket，由此可以看出网络编程主要是通过 socket 完成的。



职场点拨——同事相处之道

开篇之前，先泼一盆冷水，长期和同事在一个公司内工作，有美好的事情发生，也就会有不愉快的事情发生。假如你犯了一个错误，例如一时疏忽忘记实现一个功能，同事如果批评你，这时你应该怎么办呢？如果你总是觉得被别人指出错误是一件很没面子的事情，人家指出自己的错误，是一种耻辱，以下这些建议或能帮你克服这种心理障碍，慢慢懂得从批评中吸取教训。

1）你要从根本上明白，别人对你的批评并无损你的价值，不要以敌视的态度对待意见与你相左的同事。

2）如果别人对你的工作表现有意见，或者传出对你不满的风言风语，那说明你肯定有不对的地方，而不是故意与你做对，你要认真检讨自己，修补自己的错误。

3）不要把工作不被接受理解为自己不被接受。人无完人，每个人都会犯错误，即使你开发经验丰富，但也会有你精神不足、工作过重和承受太沉重的生活压力的时候，这时的你会很容易犯错。如果在犯错后能以正确的态度面对它，错而能改，犯错便不算什么罪大难饶的事情，反而对于你日后的工作和升职有很大好处。

第 14 章　AWT 开发窗体程序

Java 可以开发窗口程序，虽然在功能上比 Microsoft 公司推出的设计语言稍显逊色，但是仍有不少人选择使用它。Java 具有一个 AWT 包，用户可以使用这个包进行各种图形编程，AWT 是 Java 软件图形编程的工具之一，使用者非常多，任何学习 Java 程序的人都有必要掌握这个工具。本章主要内容如下：

- ❑ 什么是 AWT？
- ❑ 创建窗口。
- ❑ 创建窗口组件。
- ❑ 编写监听接口。
- ❑ 职场点拨——修炼“门面功夫”。

2010 年 XX 月 XX 日，暴雨

明天要和客户的销售经理见面，我负责为他们的产品做网络宣传。与销售经理是第一次会面，希望给对方留个好印象，我得好好琢磨下穿什么衣服。



一问一答

小菜：“走遍了很多专卖店，我终于选好了明天会见客户要穿的衣服!”

Wisdom：“会见客户当然得穿正式一些，要尽可能给对方留下好的印象。在本章的最后会介绍一些做门面功夫的技巧。”

小菜：“见客户需要注意门面，那么设计 Java 程序时，是不是也需要注意门面？”

Wisdom：“当然，在 Java 领域中，做任何项目都要考虑到界面的设计，漂亮的操作界面不但会让客户产生好感，还会给软件使用者带来愉快的心情。”

小菜：“Java 中什么能体现出界面的美观？”

Wisdom：“在桌面应用中，AWT 能体现出界面的美观，它可以用来制作窗口程序以及窗口里面的组件。”

14.1　什么是 AWT

AWT 有何用处？初学者一定要记住，AWT 可以用来制作窗口程序以及窗口里面的组件。AWT 是一个抽象的窗口工具包，是 Java 提供用来创建和设置 Java 图形界面的工具，在 AWT 包里包含了许多用来建立与平台无关的图形用户界面的类，这些类常被称做组件。抽

象窗口工具包 AWT 是与平台无关的窗口系统，同时 AWT 也是 Java 基础类 (JFC)的一部分，为 Java 程序提供图形用户界面(GUI)的标准 API。下面通过一段代码认识 AWT，其代码见“光盘：源代码/第 14 章/Chuangone1.java”：

```
import java.awt.*;//引入 AWT 包
public class Chuangone1 extends Frame
{
    //创建构造方法
    Chuangone1()
    {
        //定义窗口名称
        this.setTitle("神奇的 AWT");
        //设置窗口起始位置和大小
        this.setBounds(220,200,550,550);
        //设置窗口可见
        this.setVisible(true);
    }
    public static void main(String args[])
    {
        new Chuangone1();
    }
}
```

运行代码，得到如图 14-1 所示的结果。



图 14-1　第一个 AWT 程序

提示：AWT 程序是 Java 程序的一个包，如果要在程序中引入 AWT 的程序，必须要插入 AWT 包，插入代码格式如下:

```
import java.awt.*;
```

14.2　创建窗口

不用讲解窗口是什么东西，窗口就像一个容器，用来装置各个组件，比如创建窗口主要

是设置窗口的属性，如窗口的标题、窗口的背景颜色等都是窗口的属性，下面通过一个实例进行讲解。

实例 66：创建一个窗口

下面将给出一段程序，创建一个窗口，其代码见“光盘：源代码/第 14 章/Chuangtwo1.java”：

```
//引入 AWT 包
import java.awt.*;
public class Chuangtwo1 extends Frame
{
 //创建构造方法
 Chuangtwo1() {
     //定义窗口名称
     this.setTitle("创建窗口");
            //设置窗口背景颜色
     this.setBackground(Color.blue);
     //设置大小是否可以改变
     this.setResizable(false);
     //设置窗口是否总在最上面
     this.setAlwaysOnTop(true);
     //设置窗口起始位置和大小
     this.setBounds(100,100,400,300);
     //设置窗口可见
     this.setVisible(true);
 }
 public static void main(String args[])
{
     new Chuangtwo1();
     new Chuangtwo1();
         }
}
```

运行代码，将会得到两个窗口，如图 14-2 所示。

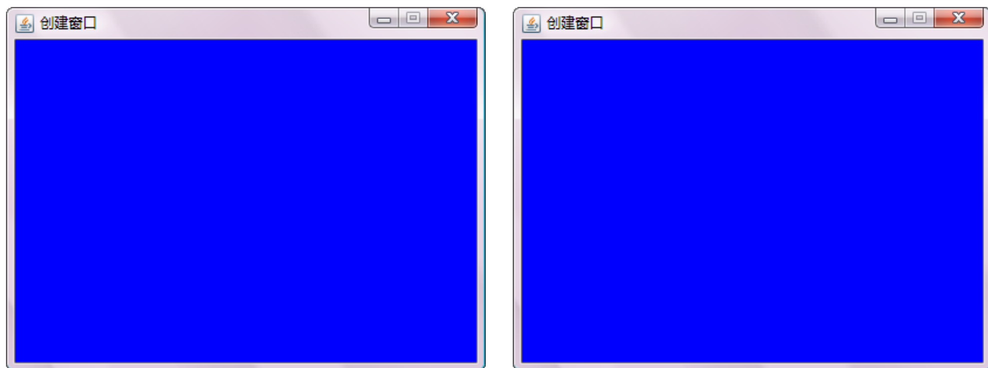


图 14-2　执行结果

提示：在窗口的右上角有关闭、最大化和最小化按钮，这些按钮目前是没有作用的，主要是没有给这些按钮编写程序。

多学一招

在上面的程序中，只要在 main 中添加一行 “new Chuangtwo1();”，就会多一个窗口。在下面这个程序中，用户也可以得出这样的结论，其代码见 “光盘：源代码/第 14 章/Chuangtwo2.java”：

```
//引入 AWT 包
import java.awt.*;
public class Chuangtwo2 extends Frame
{
 //创建构造器
 Chuangtwo2()
 {
     //定义窗口名称
     this.setTitle("夏至未至");
              //设置窗口背景颜色
     this.setBackground(Color.red);
     //设置大小是否可以改变
     this.setResizable(false);
     //设置窗口是否总在最上面
     this.setAlwaysOnTop(true);
     //设置窗口起始位置和大小，前两参数是窗口显示的位置，后两参数是窗口显示的大小
     this.setBounds(300,200,100,100);            //设置窗口可见
     this.setVisible(true);
 }

 public static void main(String args[])
 {
     new Chuangtwo2();
     new Chuangtwo2();
     new Chuangtwo2();
     new Chuangtwo2();
     new Chuangtwo2();
         }
}
```

运行代码，得到如图 14-3 所示的结果。



图 14-3　创建窗口

14.3　创建窗口组件

完成窗口创建后，用户可以向窗口中添加按钮、复选框、文本框等窗口组件，只有通过这些窗口组件，才能让窗口实现相应的功能。

实例 67：在窗口中添加组件

下面将给出一段程序，创建一个窗口，然后才窗口中添加组件，其代码见“光盘：源代码/第 14 章/Winone1.java”：

```
//引入 AWT 包
import java.awt.*;
public class Winone1 extends Frame{
 //定义组件
 TextArea ta=new TextArea("第一个组件");
 //创建构造方法
 Winone1()    {
     //定义窗口名称
     this.setTitle("在窗口中添加组件");
     //按钮添加到窗口中
     this.add(ta);
     //设置窗口背景颜色
     this.setBackground(Color.red);
     //设置大小是否可以改变
     this.setResizable(false);
     //设置窗口是否总在最上面
     this.setAlwaysOnTop(true);
     //设置窗口起始位置和大小
     this.setBounds(100,100,450,250);
     //设置窗口可见
     this.setVisible(true);
 }
 public static void main(String args[])
 {
     new Winone1();
          }
}
```

运行代码，得到如图 14-4 所示的结果。

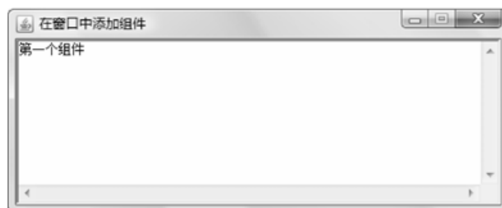
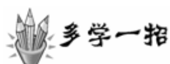


图 14-4　添加组件



在上面的程序中为窗口添加了一个文本框，下面下为窗口添加一个按钮，其代码见“光盘：源代码/第 14 章/ Winone2.java”：

```
//引入 AWT 包
import java.awt.*;
public class Winone2 extends Frame
{
 //定义组件
 Button bok=new Button("提交");
 //创建构造方法
  Winone2()
 {
     //定义窗口名称
     this.setTitle("第一个 windows 窗口");
     //把按钮添加到窗口中
     this.add(bok);
     //设置窗口背景颜色
     this.setBackground(Color.blue);
     //设置大小是否可以改变
     this.setResizable(false);
     //设置窗口是否总在最上面
     this.setAlwaysOnTop(true);
     //设置窗口起始位置和大小
     this.setBounds(100,100,450,350);
     //设置窗口可见
     this.setVisible(true);
 }
 public static void main(String args[])
 {
     new  Winone2();
         }
}
```

运行代码，得到如图 14-5 所示的结果。

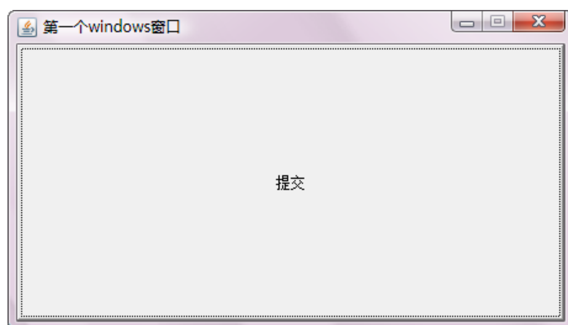


图 14-5　向 windows 添加一个按钮

14.4　布局利器

在上一节中，虽然向窗口中添加了组件，但是组件的摆放毫无规则，一个组件几乎铺满了整个窗口，这肯定不是用户所想要的。在 Java 程序设计里，有 FlowLayout、BorderLayout 和 GridLayout 等布局方式，本节将介绍把各个组件按照不同的方式进行摆放。

14.4.1　布局利器 FlowLayout

在默认的情况下，Java 的布局管理器是 FlowLayout，这个管理器会将组件按照从上到下的顺序摆放，它将所有的组件摆放在居中位置。下面通过一个实例来讲解 Flowlayout 布局管理器。

实例 68：使用 FlowLayout

下面将给出一段程序，创建一个窗口，向窗口中添加组件，然后将其摆放，其代码见“光盘：源代码/第 14 章/Wintwo1.java”：

```
//引入 AWT 包
import java.awt.*;
import java.awt.event.*;
public class Wintwo1 extends Frame
{
 //定义三个按钮组件
 Button b1=new Button("提交");
 Button b2=new Button("取消");
 Button b3=new Button("重置");
 Wintwo1()
 {
     //设置窗口名称
     this.setTitle("使用 FlowLayout 布局");
     //设置布局管理器为 FlowLayout
     this.setLayout(new FlowLayout());
     //将按钮组件放入窗口中
     this.add(b1);
     this.add(b2);
     this.add(b3);
     //设置窗口的位置和大小
     this.setBounds(100,100,450,350);

     //设置窗口的可见性
     this.setVisible(true);
 }
 public static void main(String args[])
 {
     new Wintwo1();
 }
}
```

运行代码，得到如图 14-6 所示的结果。

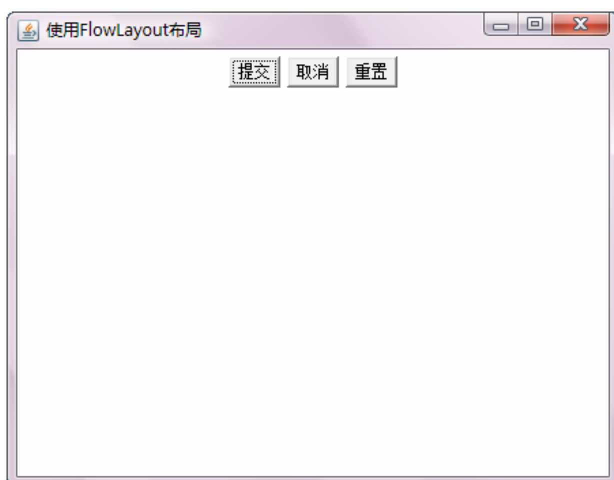


图 14-6　FlowLayout 布局管理器

多学一招

在上面的程序中向窗口中添加了按钮，然后使用 FlowLayout 为其布局，下面将给出一段代码，向窗口中添加文本框，然后再向窗口中添加三个按钮，其代码见“光盘：源代码/第 14 章/ Wintwo2.java”：

```
//引入 AWT 包
import java.awt.*;
import java.awt.event.*;
public class Wintwo2 extends Frame
{
 //定义一个文本框
 TextArea a=new TextArea("请准确填写信息");
     //定义三个按钮组件
 Button b1=new Button("提交");
 Button b2=new Button("取消");
 Button b3=new Button("重置");
 Wintwo2()
 {
     //设置窗口名称
     this.setTitle("调查信息卡");
     //设置布局管理器为 FlowLayout
     this.setLayout(new FlowLayout());
     //将按钮组件放入窗口中
     this.add(a);
     this.add(b1);
     this.add(b2);
     this.add(b3);
```

```
        //设置窗口的位置和大小
        this.setBounds(100,100,450,350);
        //设置窗口的可见性
        this.setVisible(true);
    }
    public static void main(String args[])
    {
        new Wintwo2();
    }
   }
```

运行代码，得到如图 14-7 所示的效果。

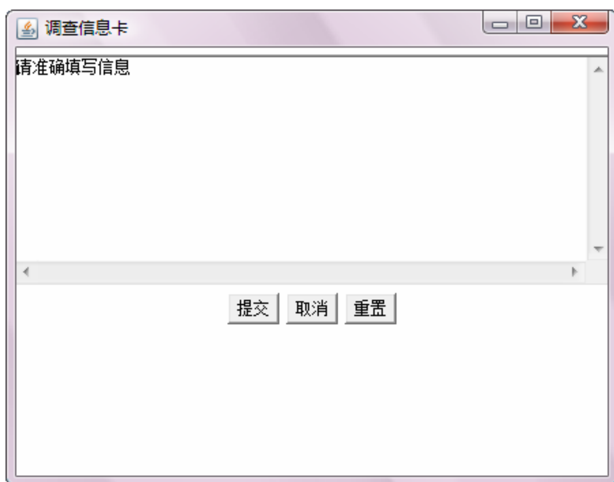


图 14-7　使用 FlowLayout

14.4.2　布局利器 BorderLayout

在 Java 程序设计中，使用 BorderLayout 布局方式可以将窗口划成 5 个区域，分别是上、下、左、右、中，下面通过一个实例进行讲解。

实例 69：使用 BorderLayout 布局

下面给出一段代码，创建一个窗口并向窗口中添加组件，然后使用 BorderLayout 布局，其代码见“光盘：源代码/第 14 章/Winthree.java”：

```
import java.awt.*;
import java.awt.event.*;
public class WinThree extends Frame
{
 //定义 5 个按钮组件
            Button b1=new Button("中间");
              Button b2=new Button("上边");
              Button b3=new Button("下边");
```

```
                Button b4=new Button("左边");
                Button b5=new Button("右边");
                WinThree()
     {

          //设置窗口名称
          this.setTitle("5 个按钮随意布局);
                //设置布局管理器为 BorderLayout
                this.setLayout(new BorderLayout());
                //将按钮组件放入窗口规定位置中
                this.add(b1,BorderLayout.CENTER);
                this.add(b2,BorderLayout.NORTH);
                this.add(b3,BorderLayout.SOUTH);
                this.add(b4,BorderLayout.WEST);
                this.add(b5,BorderLayout.EAST);
                //设置窗口的位置和大小
                this.setBounds(300,200,450,450);
                //设置窗口的可见性
                this.setVisible(true);
                //设置窗口的背景色
                this.setBackground(Color.blue);
          }
          public static void main(String args[])
          {
                new WinThree();
          }
       }
```

运行代码，得到如图 14-8 所示的结果。

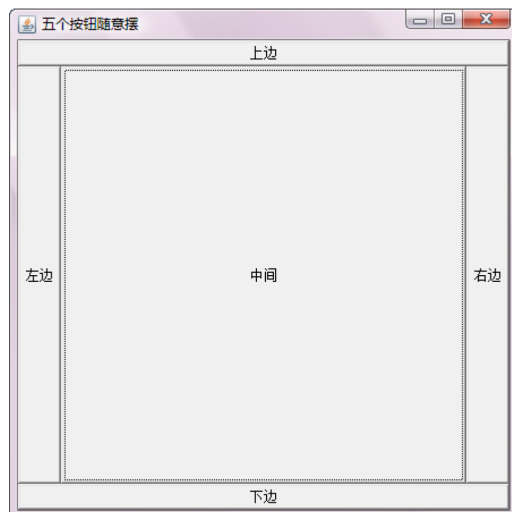
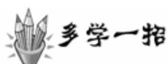


图 14-8　BorderLayout 布局



在上面的程序中向窗口中添加了按钮，然后使用 BorderLayout 为其布局，下面再给出一段代码，向窗口中添加两个文本框，然后再向窗口中添加三个按钮，其代码见“光盘：源代码/第 14 章/ Winthree1.java”：

```
import java.awt.*;
import java.awt.event.*;
public class Winthree1 extends Frame{
                //定义五个按钮组件
              TextArea b1=new TextArea("中");
             TextArea b2=new TextArea("南");
                Button b3=new Button("提交");
                Button b4=new Button("取消");
                Button b5=new Button("重置");
                Winthree1(){
        //设置窗口名称
                this.setTitle("利器 BorderLayout");
                //设置布局管理器为 BorderLayout
                this.setLayout(new BorderLayout());
                //将按钮组件放入窗口规定位置中
                this.add(b1,BorderLayout.CENTER);
                this.add(b2,BorderLayout.NORTH);
                this.add(b3,BorderLayout.SOUTH);
                this.add(b4,BorderLayout.WEST);
                this.add(b5,BorderLayout.EAST);
                //设置窗口的位置和大小
                this.setBounds(300,200,450,350);
                //设置窗口的可见性
                this.setVisible(true);  }
        public static void main(String args[]){
                new Winthree1();
        }     }
```

运行代码，得到如图 14-9 所示的结果。

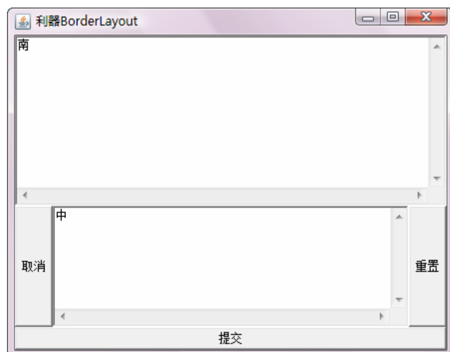


图 14-9　布局管理器 BorderLayout

14.4.3 布局利器 GridLayout

GridLayout 布局也是常用的一种布局方式，它实际上就是矩形网格，在网格中放置各个组件，每个网格的高度相等，组件会随着网格的大小而在水平方向和垂直方向拉伸，网格的大小是由容器和创建的网格的多少来确定，下面通过一个实例进行讲解。

实例 70：使用 GridLayout 布局

下面给出一段代码，创建一个窗口并向窗口中添加组件，然后使用 GridLayout 布局，其代码见“光盘：源代码/第 14 章/Winfour1.java”：

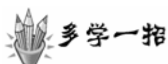
```
//引入 AWT 包
import java.awt.*;
import java.awt.event.*;
class Winfour1 extends Frame implements ActionListener
{
 int i=5;
 //定义九个按钮组件
 Button b1=new Button("按钮 A");
 Button b2=new Button("按钮 B");
 Button b3=new Button("按钮 C");
 Button b4=new Button("按钮 D");
 Button b5=new Button("按钮 E");
 Button b6=new Button("按钮 F");
 Button b7=new Button("按钮 G");
 Button b8=new Button("按钮 H");
 Button b9=new Button("按钮 I");
 Winfour1()
 {
     //设置窗口名称
     this.setTitle("布局利器 Gridlayout");
     //设置布局管理器为 2 行 3 列的 Gridlayout 布局管理器
     this.setLayout(new GridLayout(3,3));
     //将按钮组件放入窗口
     this.add(b1);
     this.add(b2);
     this.add(b3);
     this.add(b4);
     this.add(b5);
     this.add(b6);
     this.add(b7);
     this.add(b8);
     this.add(b9);
     //为每个按钮组件添加监听
     b1.addActionListener(this);
     b2.addActionListener(this);
     b3.addActionListener(this);
```

```
            b4.addActionListener(this);
            b5.addActionListener(this);
            b6.addActionListener(this);
            b7.addActionListener(this);
            b8.addActionListener(this);
            b9.addActionListener(this);
            //设置窗口的位置和大小
            this.setBounds(100,100,450,450);
            //设置窗口的可见性
            this.setVisible(true);
        }
        //实现 ActionListener 接口中的 actionPerformed 方法
        public void actionPerformed(ActionEvent e)
            {
             i++;
             Button bi=new Button("按钮"+i);
             this.add(bi);
             bi.addActionListener(this);
             this.show(true);
            }
        public static void main(String args[])
        {
             new Winfour1();
        }
       }
```

运行代码，得到如图 14-10 所示的结果。



图 14-10　使用 GridLayout 管理



在上面的程序中向窗口中添加了几个按钮，然后添加几个文本框，最后使用 GridLayout 将其布局，下面将给出一个段代码，其代码见“光盘：源代码/第 14 章/ Winfour2.java”：

```
//引入 AWT 包
import java.awt.*;
class Winfour2 extends Frame{
 //定义五个按钮组件
 TextArea b1=new TextArea("甲");
 TextArea b2=new TextArea("乙");
 Button b3=new Button("丙按钮");
 Button b4=new Button("丁按钮");
 Button b5=new Button("戊按钮");
 Winfour2()  {
     //设置窗口名称
     this.setTitle("Gridlayout 布局管理器");
     //设置布局管理器为 2 行 3 列的 Gridlayout 布局管理器
     this.setLayout(new GridLayout(2,3));
     //将按钮组件放入窗口
     this.add(b1);
     this.add(b2);
     this.add(b3);
     this.add(b4);
     this.add(b5);
     //设置窗口的位置和大小
     this.setBounds(100,100,450,450);
     //设置窗口的可见性
     this.setVisible(true);   }
 public static void main(String args[]) {
     new Winfour2();
 }}
```

运行代码，得到如图 14-11 所示的结果。



图 14-11 布局利器 GridLayout

14.4.4　布局利器 CardLayout

Cardlayout 布局管理器最大的特点是可以让一组组件中只显示某一个组件，用户可以根据需要选择需要显示的某一个组件。它提供了如下几种方法：

- first（Container p）：选择容器中的第一个组件。
- last（Container p）：选择容器中的最后一个组件。
- next（Container p）：选择容器中当前组件中下一个组件。
- previous（Container p）：选择容器中当前组件中上一个组件。
- show（Container p）：选择容器中指定的组件。

实例 71：使用 CardLayout 布局

下面将给出一段程序，创建一个窗口并在窗口添加多个组件，然后使用 CardLayout 布局，其代码见“光盘：源代码/第 14 章/Winfive1.java”：

```
//引入 AWT 包
import java.awt.*;
import java.awt.event.*;
class Winfive1 extends Frame implements ActionListener
{
 //定义一个面板
 Panel p=new Panel();
 //定义五个按钮
 Button bf=new Button("第一个");
 Button bl=new Button("最后一个");
 Button bn=new Button("下一个");
 Button bp=new Button("上一个");
 Button bg=new Button("搜索");
 //定义一个单行文本框
 TextField tf=new TextField();
 //设置 CardLayout 布局管理器
 CardLayout cl=new CardLayout();
 Winfive1()
 {
     //设置窗口名称
     this.setTitle("利用 CardLayout 布局组件");
     this.setLayout(null);
     this.add(p);
     //定义 p 面板为 CardLayout 布局管理器
     p.setLayout(cl);
     //为 CardLayout 布局管理器添加按钮
     for(int i=1;i<=3;i++)
     {
         Button btemp=new Button("布局按钮"+i);
         p.add(btemp,""+i);
     }
     //为每个组件设置大小位置并添加到容器中
```

```
            p.setBounds(10,40,100,100);
            this.add(bf);
            bf.addActionListener(this);
            bf.setBounds(120,40,60,20);
            this.add(bl);
            bl.addActionListener(this);
            bl.setBounds(120,70,60,20);
            this.add(bn);
            bn.addActionListener(this);
            bn.setBounds(120,100,60,20);
            this.add(bp);
            bp.addActionListener(this);
            bp.setBounds(120,130,60,20);
            this.add(bg);
            bg.addActionListener(this);
            bg.setBounds(60,160,40,20);
            this.add(tf);
            tf.setBounds(20,160,40,20);
            //设置窗口的位置和大小
            this.setBounds(200,200,400,380);
            //设置窗口的可见性
            this.setVisible(true);
        }
        //实现 ActionListener 接口中的 actionPerformed 方法
        public void actionPerformed(java.awt.event.ActionEvent e)
        {
            //next 方法
            if(e.getSource()==bn)
            {
                cl.next(p);
            }
            //previous 方法
            if(e.getSource()==bp)
            {
                cl.previous(p);
            }
            //first 方法
            if(e.getSource()==bf)
            {
                cl.first(p);
            }
            //last 方法
            if(e.getSource()==bl)
            {
                cl.last(p);
            }
```

```
        //show 方法
        if(e.getSource()==bg)
        {
            cl.show(p,tf.getText().trim());
            tf.setText("");
        }
    }
    public static void main(String args[])
    {
        new Winfive1();
    }
   }
```

运行代码，得到如图 14-12 所示的结果。

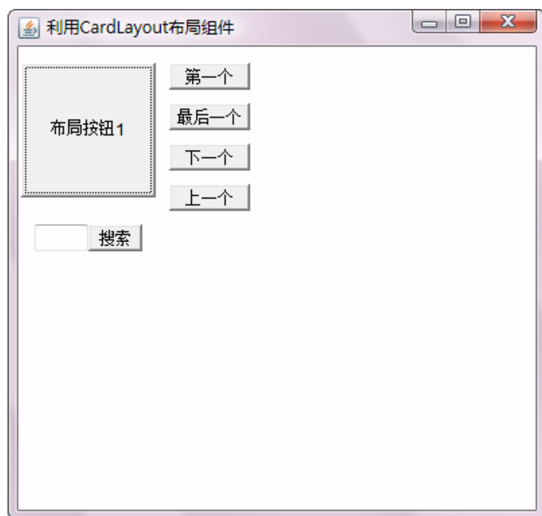


图 14-12 Cardlayout 布局管理

多学一招

在上面的布局中，Cardlayout 布局表现得十分灵活，但是 Cardlayout 布局一般不只应用于按钮组件，而是应用在各个组件上，这里只是为了让读者初步了解 Cardlayout 的布局方式。

14.4.5 布局利器 Null

在前面学了四种布局方式，但不足以应对所有情况，如下面的程序，就是前面四种布局方式都无法解决组件的布局，其代码见“光盘：源代码/第 14 章/Winnull1.java”：

```
//引入 AWT 包
import java.awt.*;
import java.awt.event.*;
```

```
class Winnull1 extends Frame
{
 //定义三个按钮
 Button b1=new Button("甲按钮");
 Button b2=new Button("乙按钮");
 Button b3=new Button("丙按钮");
 Winnull1()
 {
     //设置窗口名称
     this.setTitle("Null 布局利器");
     //将组件添加到容器
     this.add(b1);
     this.add(b2);
     this.add(b3);
     //设置组件的大小和位置
     b1.setBounds(50,50,100,50);
     b2.setBounds(50,120,100,50);
     b3.setBounds(50,190,100,50);
     //设置窗口的位置和大小
     this.setBounds(100,100,450,400);
     //设置窗口的可见性
     this.setVisible(true);
 }
 public static void main(String args[])
 {
     new Winnull1();
 }
}
```

运行代码，得到如图 14-13 所示的结果。



图 14-13　布局管理

这个结果肯定不是理想的结果，丙按钮位于中间，几乎占据了整个窗口。若要让甲按钮

和乙按钮位于丙按钮上方，就需要对代码进行修改，修改后的代码见“光盘：源代码/第 14 章/Winnull2.java”：

```
//引入 AWT 包
import java.awt.*;
import java.awt.event.*;
class Winnull2 extends Frame{
 //定义三个按钮
 Button b1=new Button("甲按钮");
 Button b2=new Button("乙按钮");
 Button b3=new Button("丙按钮");
 Winnull2(){
     //设置窗口名称
     this.setTitle("Null 布局");
     //设置 Null 布局管理器
     this.setLayout(null);
     //将组件添加到容器
     this.add(b1);
     this.add(b2);
     this.add(b3);
     //设置组件的大小和位置
     b1.setBounds(50,50,100,50);
     b2.setBounds(50,120,100,50);
     b3.setBounds(50,190,100,50);
     //设置窗口的位置和大小
     this.setBounds(100,100,450,400);
     //设置窗口的可见性
     this.setVisible(true);}
 public static void main(String args[]){
     new Winnull2();
 }
}
```

运行代码，得到如图 14-14 所示的结果。

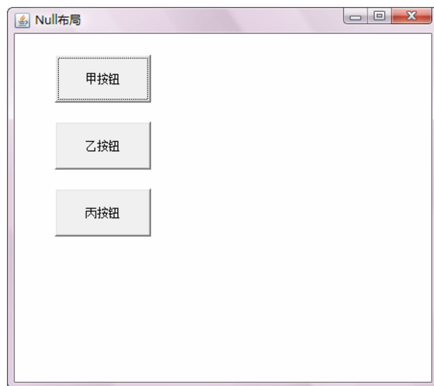


图 14-14　Null 布局

14.5 编写监听接口

监听主要是监测用户的操作从而获得用户的行为，以编写相应的方法，实现一些功能。如在前面的章节中创建的窗口以及窗口组件只是摆设，没有任何作用，要实现它们应有的功能，就需要编写监听程序和事件程序。

14.5.1 窗口监听的接口

对于窗口处理，主要是打开、关闭和最小化等操作，监听它们的接口是 WindowsListener，该接口的方法如下：

```
public void WindowsActivated（WindowsEvent e）
public void WindowsClosed（WindowsEvent e）
public void WindowsClosing（WindowsEvent e）
public void WindowsDeactivated（WindowsEvent e）
public void Windowsfied（WindowsEvent e）
public void Windowslconified（WindowsEvent e）
public void WindowsOpened（WindowsEvent e）
```

下面给出一段代码，编写一个窗口，并让窗口组件响应，其代码见“光盘：源代码/第 14 章/WindJone1.java”：

```
//引入相关
import java.awt.*;
import java.awt.event.*;
class WindJone1 extends Frame implements WindowListener
{
 WindJone1()
{
     //设置窗口名称
     this.setTitle("窗口监听");
     //为窗口添加监听
     this.addWindowListener(this);
     //设置窗口大小和位置
     this.setBounds(100,100,300,300);
     //设置窗口可见性
     this.setVisible(true);
 }
 //实现接口中的方法
 public void windowActivated(WindowEvent e)
{
     System.out.println("激活");
 }
 public void windowClosed(WindowEvent e)
{
             System.out.println("释放");
```

```
        }
        public void windowClosing(WindowEvent e)
        {
                System.out.println("菜单关闭");
            this.dispose();
        }
        public void windowDeactivated(WindowEvent e)
        {
                System.out.println("失去焦点");
        }
        public void windowDeiconified(WindowEvent e)
        {
            System.out.println("到最大化");
        }
        public void windowIconified(WindowEvent e)
        {
            System.out.println("到最小化");
        }
        public void windowOpened(WindowEvent e)
        {
            System.out.println("打开");
        }

        public static void main(String args[])
        {
                new WindJone1();
        }
    }
```

运行代码，得到如图 14-15 所示的结果。

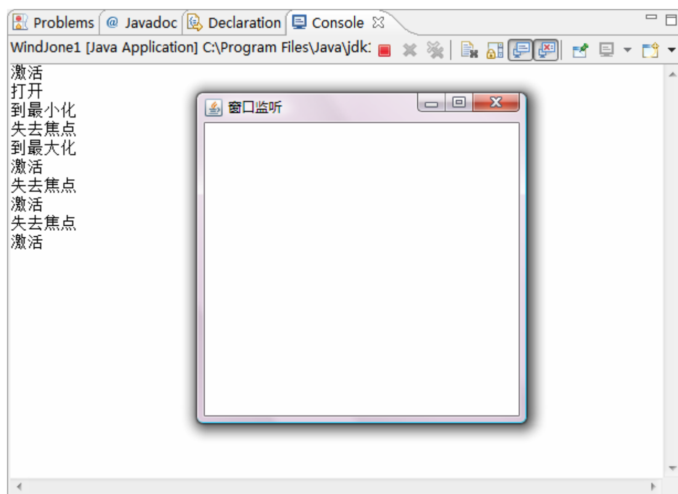


图 14-15 编写窗口监听

14.5.2　按钮监听的接口

按钮组件是任何窗口都离不开的，在前面的章节中，也讲解了许多按钮，在 Java 程序中，用户可以使用 ActionEvent 接口监听按钮组件，该接口中的方法如下：

```
public void actionPerformed(ActionEvent e);
```

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 14 章/WindJone2.java”：

```
//引入相关
import java.awt.*;
import java.awt.event.*;
class WindJone2 extends Frame implements ActionListener
{
 //定义两个按钮
 Button b1=new Button("确定");
 Button b2=new Button("退出");
WindJone2()
 {
     //设置窗口名称
     this.setTitle("进入系统");
     //将按钮添加到窗口中
     this.add(b1);
     this.add(b2,BorderLayout.SOUTH);
     //为按钮添加监听
     b1.addActionListener(this);
     b2.addActionListener(this);
     //设置窗口位置和大小
     this.setBounds(100,100,300,300);
     //设置窗口可见性
     this.setVisible(true);
 }
 //实现 ActionListener 接口中的方法
 public void actionPerformed(ActionEvent e)
 {
     if(e.getSource()==b1)
     {
             System.out.println("进入系统");
     }
     else
     {
             System.out.println("退出");
     }
 }
    public static void main(String args[])
    {
```

```
            new WindJone2();
        }
    }
```

运行代码，得到如图 14-16 所示的效果。

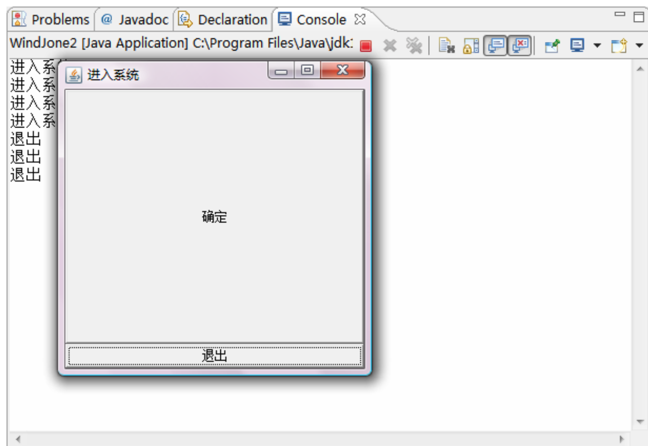


图 14-16 监听按钮

14.5.3 文本框监听的接口

在 Java 程序设计中，监听文本框是比较常见的监听方法，许多 Java 窗口程序都离不开文本框组件的监听。Java 程序为监听文本框提供了 TextListener 接口，主要功能是确定文本值何时改变，其格式如下：

```
public void textValueChangged（TextEvent e）;
```

实例 72：使用文本框监听的接口

下面将给出一段代码，创建一个窗口并在窗口添加文本框组件，然后为其文本框编写事件，其代码见“光盘：源代码/第 14 章/wintext.java”：

```
//引入相关
import java.awt.*;
import java.awt.event.*;
class wintext extends Frame implements TextListener
{
 //定义一个文本组件
 TextArea ta=new TextArea("请输入心中想说的话");

 wintext()
 {
     //设置窗口名称
     this.setTitle("监听文本框");
     //将文本组件添加到窗口中
```

```
        this.add(ta);
        //设置文字样式
        Font f=new Font("楷体",Font.ITALIC+Font.BOLD,18);
        //将文字添加到文本组件中
        ta.setFont(f);
        //为文本组件添加监听
        ta.addTextListener(this);
        //设置文本组件内内容颜色
        ta.setForeground(Color.blue);
        //设置窗口大小和位置
        this.setBounds(100,100,450,400);
        //设置窗口可见性
        this.setVisible(true);
    }

    //实现接口中的方法
    public void textValueChanged(TextEvent e)
    {
        System.out.println(ta.getText());
    }

    public static void main(String args[])
    {
        new wintext();
    }
  }
```

运行代码，得到如图 14-17 所示的结果。

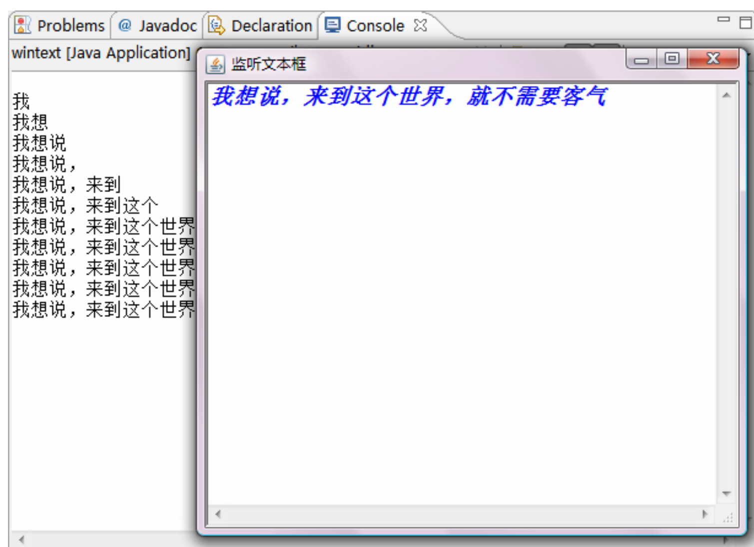
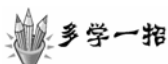


图 14-17　执行结果



在上面的监听中，使用了方法监听文本框。关于窗口的属性，在注释中都有讲解，读者可以试着去改变一些属性值，以调整窗口各个组件的属性。下面再展示一段代码，是修改上面一段代码得来的，执行结果大同小异，其代码见“光盘：源代码/第 14 章/Wintext2.java”：

```
//引入相关
import java.awt.*;
import java.awt.event.*;
class Wintext2 extends Frame implements TextListener
{
 //定义一个文本组件
 TextArea ta=new TextArea("输入文字");
 Wintext2()
 {
     //设置窗口名称
     this.setTitle("在文本框中输入文字");
     //将文本组件添加到窗口中
     this.add(ta);
     //设置文字样式
     Font f=new Font("黑体",Font.ITALIC+Font.BOLD,18);
     //将文字添加到文本组件中
     ta.setFont(f);
     //为文本组件添加监听
     ta.addTextListener(this);
     //设置文本组件内内容颜色
     ta.setForeground(Color.red);
     //设置窗口大小和位置
     this.setBounds(100,100,530,380);
     //设置窗口可见性
     this.setVisible(true);
 }
 //实现接口中的方法
 public void textValueChanged(TextEvent e) {
     System.out.println(ta.getText());   }
 public static void main(String args[]){
     new Wintext2();     }}
```

14.6　疑难问题解析

本章详细介绍了什么是 AWT 窗口、如何给 AWT 窗口添加组件，以及 AWT 的布局的基础知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：在 Java 程序设计里，AWT 是创建窗口的唯一方法吗？

解答：Java 程序设计里，AWT 不是创建窗口的唯一方法，但是任何一种创建窗口的方法，都是以 AWT 为基础，因此用户只有熟练掌握 AWT 的原理与功能，才能用好其他创建窗口的工具。

读者疑问：在窗口程序的编写过程中，初学者应该注意什么？

解答：各个环节都应该主意，首先是要设计出漂亮的界面，然后是为窗口中的各个组件编写监听接口，再为它编写方法，最后是为窗口各个组件编写程序，让它实现功能。



职场点拨——修炼“门面功夫”

在工作中我们需要特别注意自身形象，虽然个性的衣着会引起大家的关注，但是还是建议应该以大众为主，要跟随整个公司的企业文化。文化氛围决定了公司的着装风格，例如国企要求职员踏实工作，对职员的着装要求简洁大方，如果衣着过于正式或者华丽，都将令领导及同事产生紧张感。

对于职场新人来说，在接到录用通知时，一定要详细咨询关于着装的问题，如果公司对着装没有要求，员工们穿着也都很随意，你大可放心。无论是面试时还是职场亮相时，在你小心谨慎遵守各项规则之前，别忘了将个性的东西加进去，这样的你才有机会在职场把握有度，自由成长。

为了让读者们更好地对着装做出选择，接下来提出如下 4 点建议供大家参考。

（1）个体性

着装的个性有两层意思，第一，是着装应当照顾自身的特点，“量体裁衣”，使之适应自身，并扬长避短。第二，是着装应创造并保持自己独有的风格，在允许的前提下，着装在某些方面应当与众不同。

（2）整体性

着装要坚持整体性，重点是要注意搭配，主要表现在两个方面，其一，是要恪守服装本身约定俗成的搭配。例如，穿西装时应配皮鞋，而不能穿布鞋、凉鞋和运动鞋。其二，是要使服装各个部分相互适应，局部服从于整体，力求展现着装的整体之美。

（3）整洁性

不管在什么情况下，着装都要力求整洁。

（4）技巧性

不同的服装，有不同的搭配和约定俗成的穿法。例如当穿单排扣西装上衣时，两粒纽扣要系上面一粒，三粒要系中间一粒或是上面两粒。女士穿裙子时，所穿丝袜应被裙子下摆所遮掩，而不宜露于裙摆之外。

第 15 章　深入 Java 窗口编程

随着时代的发展，AWT 已经不能满足程序设计者的需求，这时候一个华丽的编程工具横空出世，它就是 Swing。Swing 是建立在 AWT 基础之上的，在不同平台都能保持组件的界面样式，因此得到了非常广泛在应用，在本章将会详细讲解 Swing 的使用。本章主要内容如下：

- Swing 的开发步骤。
- Icon 接口。
- 添加组件。
- 布局管理器。
- 职场点拨——今天你想跳槽了吗？

2010 年 XX 月 XX 日，小雪

我已经工作八个月了，工资从当初的 3000 元涨到了 4000 元，但依然感觉不够花，女友还提起买房结婚……我感觉压力好大，真想换一家单位寻求更高的待遇。



一问一答

小菜："我想寻找更好的发展，现在这家公司的待遇太低了！"

Wisdom："你可得考虑好了，人往高处走这无可厚非，但是你能确保一定能找到更好待遇的工作吗？"

小菜："这个……"

Wisdom："你也不能确定吧，建议你看一下本章最后的跳槽经验谈，或许对你会有帮助。"

小菜："我接触过的 Java 程序基本都是在 Web 领域，Java 在窗口方面的表现怎么样？"

Wisdom："桌面应用一直不是 Java 的强项，虽然在大型工程里面，Java 很少见，但是在一般操作系统和游戏里 Java 几乎无处不在，所以说学会用 Java 编写窗口程序也是很有必要的。"

15.1　Swing 的开发步骤

Swing 是一个用于 Java 程序界面的开发工具包，它以 AWT 为基础，可以使用任何插件的外观风格。Swing 开发人员只用很少的代码就可以利用 Swing 丰富、灵活的功能和模块化

组件来创建优秀的用户界面，开发 Swing 界面的主要步骤是导入 Swing 包、选择风格界面、设置顶层容器、设置按钮和标签、将组建添加到容器上、为组件设置步骤，最后处理事件，如图 15-1 所示。

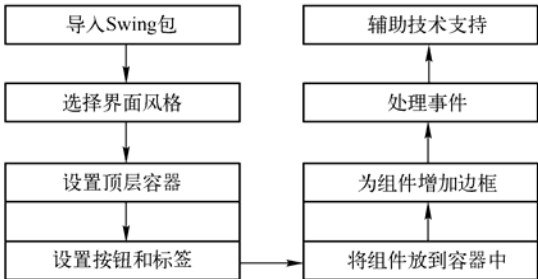


图 15-1　Swing 开发步骤

15.2　创建窗口

Swing 的窗口和其他的没有什么区别，首先要创建窗体，有标题、最小化按钮、最大化按钮和关闭等，和 AWT 十分类似。

15.2.1　JFrame 简介和方法

使用 Swing 开发窗口一般都会用到 JFrame 类，Swing 中的 JFrame 类和上一章讲解的 Frame 类十分相似，它是顶层的 Swing 容器。如果要创建窗体，通常需要新建一个类来继承它，使窗体中有边界、标题栏、最小化、最大化、关闭按钮等。JFrame 类中有一个 SetDefaultCloseOperation（int operation）方法，其方法介绍如下：

- DO_NOTHING_ON_CLOSE（单击无效）。
- HIDE_ON_CLOSE（单击后窗口隐藏）。
- HIDE_ON_CLOSE（单击后窗口释放）。
- EXIT_ON_CLOSE（单击后退出）。

15.2.2　创建第一个 Swing 窗口

熟悉了创建 Swing 的方法后，下面将通过一个实例讲解如何创建一个 Swing 窗口。

实例 73：使用 Swing 窗口和 JFrame 窗口

下面通过代码创建一个 Swing 窗口，让读者更进一步地认识 Swing 窗口和 JFrame 窗口，其代码见“光盘：源代码/第 15 章/swingone.java”：

```
 //引入相关包
 import java.awt.*;
import java.awt.event.*;
 import javax.swing.*;
 public class swingone extends JFrame
//继承 JFrame 类
```

```
    {
        public swingone() {
            //设置窗口名称
            this.setTitle("第一个 Swing 窗口");
            //设置一个 GridLayout 布局管理器
            this.setLayout(new GridLayout(4,4));
            //设置窗口位置和大小
            this.setBounds(10,10,400,400);
            //设置窗口可见
            this.setVisible(true);
            //设置当点击窗口的关闭按钮时退出
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
        public static void main(String args[])
        {
            new swingone();
        }
    }
```

运行代码，得到如图 15-2 所示的结果。

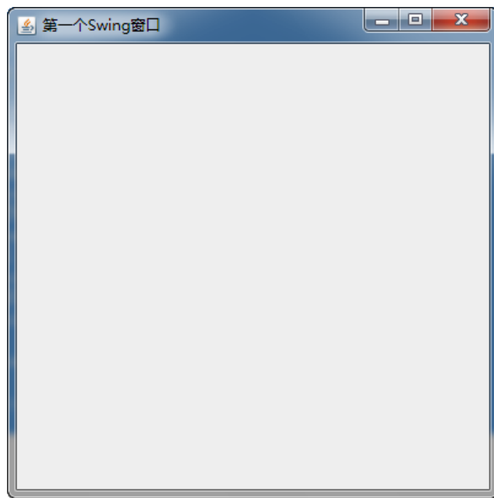


图 15-2　Swing 窗口

多学一招

前面讲过，若要使窗口的按钮有作用，必须要编写一定的方法程序。Swing 也不例外，不过 Swing 的操作十分简单，只需要一行代码就可以了。在上面这段代码中，标题的按钮是有作用的，这与 AWT 有着很大的区别。下面通过改写上面实例的代码，加深读者对 Swing 的理解。修改后的代码见“光盘：源代码/第 15 章/swingone1.java”：

```
//引入相关包
import java.awt.*;
```

```
import java.awt.event.*;
import javax.swing.*;
public class Swingone1 extends JFrame//继承 JFrame 类
{
 public Swingone1()
 {
     //设置窗口名称
     this.setTitle("Swing 窗口与 AWT 窗口的异同");
   //设置窗口是否总在最上面
     this.setAlwaysOnTop(true);
     //设置一个 GridLayout 布局管理器
     this.setLayout(new GridLayout(2,2));
     //设置窗口位置和大小
     this.setBounds(10,10,600,500);
     //设置窗口可见
     this.setVisible(true);
     //设置当点击窗口的关闭按钮时退出
     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }
 public static void main(String args[]) {
     new Swingone1();
 }
}
```

运行代码，得到如图 15-3 所示的结果。

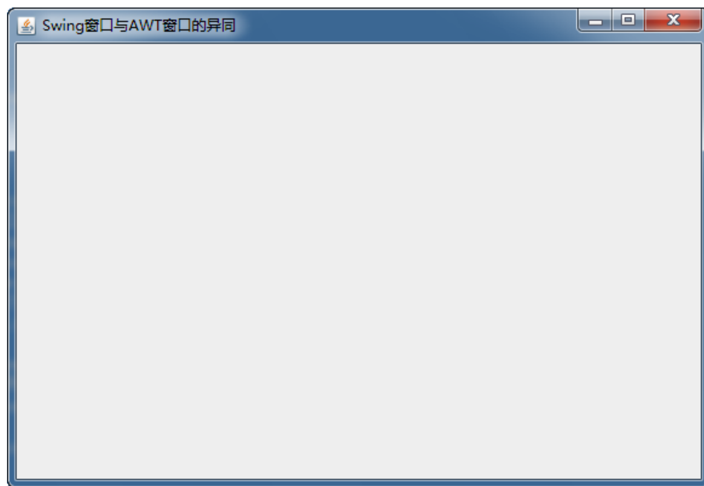


图 15-3　Swing 窗口

15.3　Icon 接口

Icon 对于 Swing 来说是一个十分重要的接口，通过该程序可以创建简单的几何图形，也

可以调用外部的图片，从而起到美化窗口的作用。

用来创建 Icon 对象的方法有三种，其介绍如下：

- new ImageIcon（Images i）。
- new ImageIcon（String fileImages ）（参数为图像的名称）。
- new ImageIcon（URL u）（参数为网络地址）。

Icon 接口的方法也有三种，其介绍如下：

- paintIcon（Graphics）。
- getIconWidth（设置图形宽度）。
- getIconHeight(设置图形长度)。

在学习如何添加图像前，还需要学习一个 JLabel 类。JLabel 是一种既可以包含文本，又可以包含图像的控件，但该控件不能影响用户的动作。创建 JLabel 非常简单，格式如下：

```
JLabel T1=new JLabel（”a”）;
```

这样就创建了一个 JLabel 控件，然后通过 add（）方法将其添加到容器中，添加方法如下：

```
this.add(T1);
```

实例 74：使用 Icon 接口

下面通过代码创建一个 Swing 窗口，然后实现 Icon 接口，让读者认识 Icon 接口，其代码见“光盘：源代码/第 15 章/swingicon.java”：

```
//引入相关包
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class swingicon extends JFrame//继承 JFrame 类{
 //定义两个标签
 JLabel j1=new JLabel("A 文字");
 JLabel j2=new JLabel("B 文字");
 public swingicon()     {
     //设置一个 GridLayout 布局管理器
     this.setLayout(new GridLayout(2,2));
     //把标签加到布局管理器中
     this.add(j1);
     this.add(j2);
     //引入原有图片
     ImageIcon i1=new ImageIcon("1.jpg");
     j1.setIcon(i1);
     //引入自做图片
     Icon i2=new MyIconImp();
     j2.setIcon(i2);
     //设置窗口名称
     this.setTitle("Icon 接口在 Swing 的应用");
     //设置窗口是否总在最上面
     this.setAlwaysOnTop(true);
```

```
            //设置窗口位置和大小
            this.setBounds(10,10,700,600);
            //设置窗口可见
            this.setVisible(true);
            //设置当点击窗口的关闭按钮时退出
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
            public static void main(String args[])   {
            new swingicon();
        }}
       //定义一个绘制图片的类
       class MyIconImp implements Icon{
        //设置图片宽度
        public int getIconWidth()
        {
            return 200;
        }
        //设置图片长度
        public int getIconHeight()   {
            return 200;
        }
        //设置图片
        public void paintIcon(Component c,Graphics g,int x,int y)    {
            for(int i=0;i<6;i++){
                g.setColor(new Color(10*i,5*i,3*i));
                g.fillOval(5,15+35*i,60,80);
            }
        }}
```

运行代码，得到如图 15-4 所示的效果。

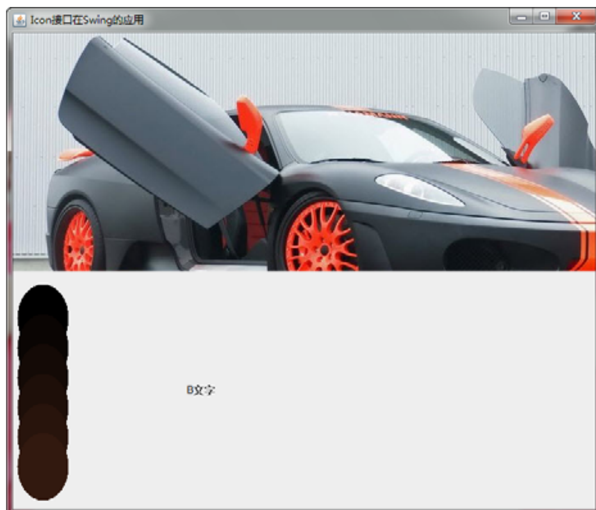
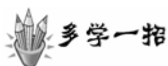


图 15-4　Icon 接口



在图像编程时，一定要注意设计的美感，若要设计背景，就需要使用淡雅的颜色和图片。若要作为主要显示的元素，则要使用对比较鲜明的颜色。除黑色和白色外，一个窗口的颜色最多不能大于三种，这是美感的基础。

15.4　添加组件

在 Swing 窗口中常需要使用不同的方法添加不同的窗口组件，如按钮、文本框等来实现程序交互功能。在本节中将详细讲解 Swing 窗口组件的添加。

15.4.1　弹出式菜单

在开篇功能中，已经见识过弹出式菜单，弹出式菜单是十分常见的组件之一，在 Swing 窗口中，可以使用 JPopupMenu 类定义弹出式菜单，它有两种常用方法，其介绍如下：

- Add(JMenuItem e)：向菜单中增加菜单项。
- Show()：显示菜单。

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/swingtan.java”：

```
//引入相关包
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class swingtan extends JFrame//继承 JFrame 类
{
 //定义一个弹出式菜单
 JPopupMenu j=new JPopupMenu();
 public swingtan ()
 {
     //设置窗口名称
     this.setTitle("弹出菜单");
     //设置一个 GridLayout 布局管理器
     this.setLayout(new GridLayout(2,2));
     //设置窗口位置和大小
     this.setBounds(10,10,600,400);
     //设置窗口可见
     this.setVisible(true);
     //设置当点击窗口的关闭按钮时退出
     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     //引入 jiaPopMenu()方法
     this.jiaPopMenu();
 }
 public void jiaPopMenu()
 {
     //定义菜单项
```

```
        JMenuItem jmi=new JMenuItem("剪切");
        //为 A 添加监听事件
        jmi.addActionListener(
            new ActionListener()
            {
                public void actionPerformed (ActionEvent e)
                {
                     System.out.println("正在剪 切!!!");
                }
            }
            );
        //将弹出添加到菜单
        j.add(jmi);
        //定义菜单项
        jmi=new JMenuItem("复制");
        jmi.addActionListener(
                new ActionListener()
                {
                    public void actionPerformed
    (ActionEvent e)
                    {
                         System.out.println("正在复 制!!!");
                    }
                }
                );
      //将弹出添加到菜单
         j.add(jmi);
         //定义菜单项
         jmi=new JMenuItem("粘贴");
         jmi.addActionListener(
                 new ActionListener()
                 {
                     public void actionPerformed
    (ActionEvent e)
                     {
                          System.out.println("正在粘贴!!!");
                     }
                  }
                  );
         //将弹出添加到菜单
         j.add(jmi);
         //定义菜单项
         jmi=new JMenuItem("退出");
         jmi.addActionListener(
                 new ActionListener()
                 {
                     public void actionPerformed  (ActionEvent e)
                     {
```

```
                              System.out.println("正在退出!!!");
                         }
                     }
                     );
               //将弹出添加到菜单
               j.add(jmi);
               j.add(jmi);
               //编写右键弹出点击事件
                   this.addMouseListener(
                  new MouseAdapter()
                  {
                     public void mouseReleased(MouseEvent e)
                     {
                         if(e.isPopupTrigger())
                         {
                               j.show(e.getComponent  (),e.getX(),e.getY());
                         }
                      }
                   }
                 );
         }
         public static void main(String args[])
         {
            new swingtan ();
         }
      }
```

运行代码，得到如图 15-5 所示的结果。

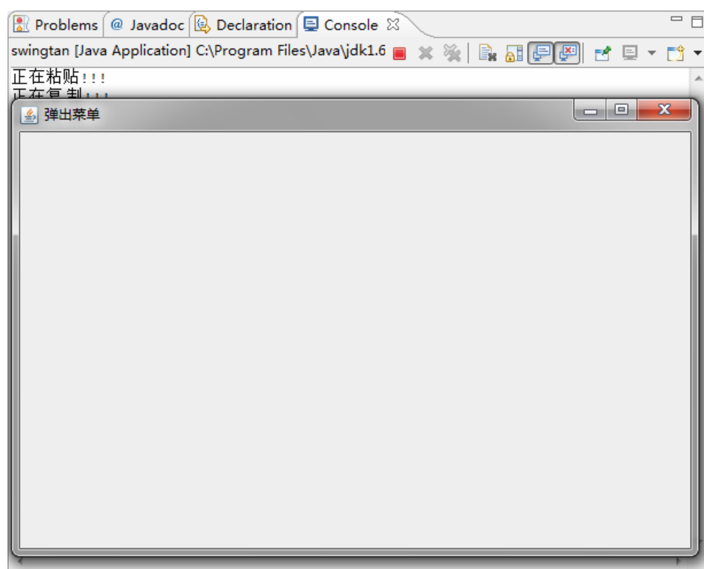


图 15-5　执行结果

15.4.2 文本框

文本框是用来收集用户信息的重要工具，Java 为 Swing 提供了 JtextField 类，用户可以通过查看 API 手册了解它的构造方法，这里只通过一个实例代码为读者做简单介绍。

实例 75：在 Swing 窗口中添加一个文本框组件

下面通过代码创建一个 Swing 窗口，并添加一个文本框组件，其代码见“光盘：源代码/第 15 章/Swingtext.java”：

```
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Toolkit;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
public class Swingtext
 {
 private JLabel usernameLabel;
 private JTextField usernameTextField;
 private JLabel answerLabel;
 public void add(JFrame frame) {
 frame.setTitle("登录");
 frame.setLayout(null);
 usernameLabel = new JLabel("用户名");
 usernameLabel.setBounds(60, 90, 70, 30);
 usernameLabel.setFont(new Font("黑体", Font.BOLD, 16));
 usernameTextField = new JTextField();
 usernameTextField.setBounds(140, 90, 120, 30);
 usernameTextField.setFont(new Font("宋体", Font.BOLD, 16));
// 设置文本框内容的字体样式
 usernameTextField.setHorizontalAlignment(JTextField.CENTER);
// 设置文本框内容的水平对齐方式
    usernameTextField.addKeyListener(new KeyListener() {
        public void keyPressed(KeyEvent e)
{
// 捕获按键被按下的事件
        }
        public void keyTyped(KeyEvent e)
{
        }
        public void keyReleased(KeyEvent e)
{
// 捕获按键被释的事件
        JTextField usernameTextField = (JTextField) e.getSource();
 answerLabel.setText("您输入的网名是：“" + usernameTextField.getText()
+"”，请检查！");
```

```
            }
        });
        answerLabel = new JLabel();
        answerLabel.setBounds(60, 130, 300, 30);
        answerLabel.setFont(new Font("汉真广标", Font.BOLD, 16));
        frame.add(usernameLabel);
        frame.add(usernameTextField);
        frame.add(answerLabel);
    }

    public static void main(String[] args) {
    // 创建指定标题的 JFrame 窗口对象
    JFrame frame = new JFrame("利用 JFrame 创建窗口");
    // 关闭按钮的动作为退出窗口
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400, 300);// 设置窗口大小
    // 获得显示器大小对象
    Dimension displaySize = Toolkit.getDefaultToolkit().getScreenSize();
    // 获得窗口大小对象
    Dimension frameSize = frame.getSize();
        if (frameSize.width > displaySize.width)
        // 窗口的宽度不能大于显示器的宽度
        frameSize.width = displaySize.width;
    // 窗口的高度不能大于显示器的高度
        if (frameSize.height > displaySize.height)
     frameSize.height = displaySize.height;
        frame.setLocation((displaySize.width - frameSize.width) / 2,
    // 设置窗口居中显示器显示
     (displaySize.height - frameSize.height) / 2);
        Swingtext SwingWen = new Swingtext();
        // 向 JFrame 窗口添加标签
    SwingWen.add(frame) ;// 设置窗口为可见的，默认为不可见
        frame.setVisible(true) ;
    }
    }
```

运行代码，得到如图 15-6 所示的结果。

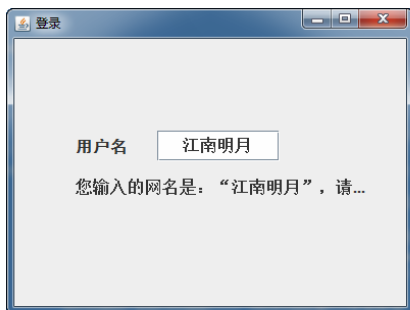
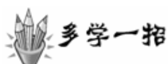


图 15-6　执行结果



实例中的文本框在输入保密内容时实用性较差，有一种特殊的文本框是专门用来输入保密信息的文本框，也称密码框。下面通过一段代码进行详细讲解，其代码见“光盘：源代码/第 15 章/swingpass.java”：

```
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPasswordField;
public class Swingpass
 {
 private JLabel passwordLabel;
 private JPasswordField passwordField;
 public void add(JFrame frame)
{
 frame.setTitle("请输入密码框");
     frame.setLayout(null);
     passwordLabel = new JLabel("密码：");
     passwordLabel.setBounds(125, 120, 40, 20);
     passwordField = new JPasswordField();
     passwordField.setBounds(175, 120, 100, 20);
     passwordField.setEchoChar('*');
     frame.add(passwordLabel);
     frame.add(passwordField);
 }
 public static void main(String[] args)
 {
// 创建指定标题的 JFrame 窗口对象
 JFrame frame = new JFrame("利用 JFrame 创建窗口");
// 关闭按钮的动作为退出窗口
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 // 设置窗口大小
 frame.setSize(500, 500);
// 获得显示器大小对象
 Dimension displaySize = Toolkit.getDefaultToolkit().getScreenSize();
// 获得窗口大小对象
     Dimension frameSize = frame.getSize();
     if (frameSize.width > displaySize.width)
// 窗口的宽度不能大于显示器的宽度
     frameSize.width = displaySize.width;
     if (frameSize.height > displaySize.height)
// 窗口的高度不能大于显示器的高度
 frameSize.height = displaySize.height;
 frame.setLocation((displaySize.width - frameSize.width) / 2,
```

```
        // 设置窗口居中显示器显示
         (displaySize.height - frameSize.height) / 2);
         // 向 JFrame 窗口添加标签
             Swingpass SwingPassword = new Swingpass();
             SwingPassword.add(frame);
         // 设置窗口为可见的，默认为不可见
             frame.setVisible(true);
         }
        }
```

运行代码，得到如图 15-7 所示的结果。

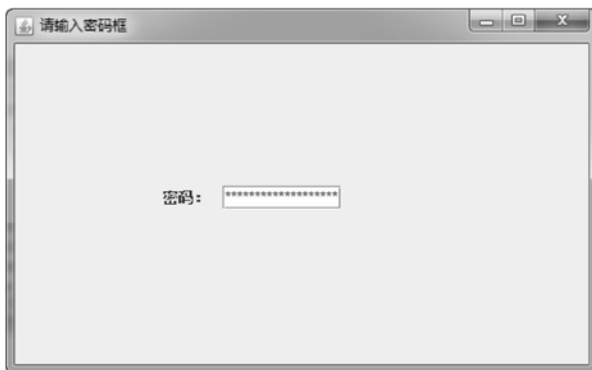


图 15-7　输入的密码

15.4.3　菜单

菜单是窗口的主题，也是窗口的导航器，它能够将窗口的信息清楚地分类，方便用户进行操作。在 Swing 中，用户可以通过 JMenu 类、JMenuItem 类和 JMenuBar 类组件创建菜单，这里只介绍基础的 JMenuBar 类，读者可以通过查看 Java API 手册了解其他制作方法，这里不再赘述。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/Swingcai.java”：

```
        //引入相关包
        import java.awt.*;
        import java.awt.event.*;
        import javax.swing.*;
        import java.util.*;
        import javax.swing.event.*;
        public class Swingcai extends JFrame
        {
            //定义菜单条组件
            JMenuBar jmb=new JMenuBar();
            //定义三个菜单组件
            JMenu cai1=new JMenu("文件");
            JMenu cai2=new JMenu("编辑");
            JMenu cai4=new JMenu("帮助");
```

```
        JMenu cai3=new JMenu("新建");
        //定义三个菜单项组件
        JMenuItem cail1=new JMenuItem("文本文件");
        JMenuItem cail2=new JMenuItem("复制");
        JMenuItem cail3=new JMenuItem("剪切");
        JMenuItem cail4=new JMenuItem("粘贴");
        public Swingcai()
    {
         //设置一个 GridLayout 布局管理器
         this.setLayout(new GridLayout(1,1));
         //调整组件间的包含关系
            jmb.add(cai1);
          jmb.add(cai2);
          cai1.add(cai3);
          jmb.add(cai4);
          cai3.add(cail1);
          cai2.add(cail2);
          cai2.add(cail3);
          cai2.add(cail4);
          //设置窗口名称
         this.setTitle("编辑文档");
          this.setJMenuBar(jmb);
         //设置窗口位置和大小
         this.setBounds(10,10,600,400);
         //设置窗口可见
         this.setVisible(true);
         //设置当点击窗口的关闭按钮时退出
         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String args[]) {
         new Swingcai();
    }
   }
```

运行代码，得到如图 15-8 所示的结果。



图 15-8　菜单

15.4.4　单选按钮

单选按钮是一个十分方便的组件，如在网页中输入性别信息，常常是以单选按钮的形式出现的，如果用文本框统计信息，则会很麻烦。用户可以使用 ButtonGroup 类和 JRadionButton 类创建单选按钮。ButtonGroup 的方法介绍如下：

- add（AbstractButton b）：添加按钮到按钮组中。
- remove（AbstractButton b）：从按钮组中移除按钮。
- getButton()：返回按钮组中包含按钮的个数，返回值为 int 型。
- getElements()：返回一个 Enumeration 类型的对象，通过其可以遍历按钮组中包含的所有对象。

JRadioButton 类常用的构造方法介绍如下：

- JRadioButton()：创建一个默认的单选按钮，默认情况即未指定文本，未指定图像，并且未被选择。
- JRadioButton(String text)：创建一个指定文本的单选按钮。
- JRadioButton(String text，boolean selected)：创建一个指定文本和选择状态的单选按钮。

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/swingDan.java”：

```
import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Enumeration;
import javax.swing.AbstractButton;
import javax.swing.ButtonGroup;
import javax.swing.ButtonModel;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JRadioButton;
public class SwingDan {
 private JLabel sexLabel;
 private JRadioButton manRadioButton;
 private JRadioButton womanRadioButton;
 private JLabel answerLabel;
 public void add(JFrame frame) {
     frame.setTitle("输入性别信息");
     frame.setLayout(null);
     sexLabel = new JLabel("性别：");
     sexLabel.setBounds(120, 100, 40, 20);
// 创建指定文本和选择状态的单选按钮对象
     manRadioButton = new JRadioButton("男孩", true);
manRadioButton.setBounds(170, 100, 50, 20);
// 捕获单选按钮被选中的事件
manRadioButton.addActionListener(new ActionListener()
```

```
        {
        public void actionPerformed(ActionEvent e) {
                    answerLabel.setText("您选中的单选按钮是："
        // 设置提示标签的内容
                    + manRadioButton.getText());
                            }
                        });
        // 创建指定文本的单选按钮对象
             womanRadioButton=new JRadioButton("女孩");
             womanRadioButton.setBounds(230, 100, 50, 20);
             womanRadioButton.addActionListener(new ActionListener()
         {// 捕获单选按钮被选中的事件
                           public void actionPerformed(ActionEvent e)
         {
        // 设置提示标签的内容
             answerLabel.setText("您选中的单选按钮是："
                                         + womanRadioButton.getText());
                           }
                      });
        // 创建一个选按钮组对象
        ButtonGroup sexRadioButtonGroup = new ButtonGroup();
        // 将单选按钮对象添加到按钮组对象中
        sexRadioButtonGroup.add(manRadioButton);
        // 将单选按钮对象添加到按对象中
        sexRadioButtonGroup.add(womanRadioButton);
            answerLabel=new JLabel();
            answerLabel.setBounds(120, 140, 200, 20);
            Enumeration<AbstractButton> elements = sexRadioButtonGroup
        // 遍历按钮组中的所有按钮
                      .getElements();
            while (elements.hasMoreElements())
         {
        // 设置提示标签的默认内容
                 AbstractButton button = elements.nextElement();
                 if (button.isSelected())
            answerLabel.setText("默认选中的单选按钮是：" + button.getText());
            }
            frame.add(sexLabel);
        // 将单选按钮添加到 JFrame 窗口中
            frame.add(manRadioButton);
        // 将单选按钮添加到 JFrame 窗口中
            frame.add(womanRadioButton);
            frame.add(answerLabel);
         }
         public static void main(String[] args)
         {
```

```
        // 创建指定标题的 JFrame 窗口对象
         JFrame frame=new JFrame("利用 JFrame 创建窗口");
        // 关闭按钮的动作为退出窗口
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // 设置窗口大小
             frame.setSize(400, 300);
        // 获得显示器大小对象
              Dimension displaySize = Toolkit.getDefaultToolkit().getScreenSize();
              // 获得窗口大小对象
        Dimension frameSize = frame.getSize();
              if (frameSize.width > displaySize.width)
        // 窗口的宽度不能大于显示器的宽度
              frameSize.width = displaySize.width;
              if (frameSize.height > displaySize.height)
        // 窗口的高度不能大于显示器的高度
           frameSize.height=displaySize.height;
        // 设置窗口居中显示器显示
              frame.setLocation((displaySize.width - frameSize.width) / 2,
           displaySize.height-frameSize.height/ 2);
              SwingDan SwingDan1=new SwingDan();
        // 向 JFrame 窗口添加标签
              SwingDan1.add(frame);
        // 设置窗口为可见的，默认为不可见
              frame.setVisible(true);
         }
        }
```

运行代码，得到如图 15-9 所示结果。



图 15-9　单选按钮

15.4.5　复选框按钮

复选框一般是用来收集可以多项选择的表单组件，如爱好等元素，用户可以通过 JCheck 类定义复选框。复选框和单选按钮十分相似，只是复选框可以有任意数量的复选项被选中。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/Swingfu.java”：

```
import java.awt.*;
import javax.swing.*;
public class Swingfu extends JFrame
{
 //定义复选框
 JCheckBox jcb=new JCheckBox("新加坡");
 JCheckBox jcb1=new JCheckBox("澳大利亚");
 JCheckBox jcb2=new JCheckBox("美国");
 JCheckBox jcb3=new JCheckBox("中国");
 JCheckBox jcb4=new JCheckBox("俄罗斯");
 JCheckBox jcb5=new JCheckBox("印度");
 public Swingfu()
 {
      //设置窗口名称
      this.setTitle("你喜欢哪个国家");
      //设置一个 GridLayout 布局管理器
      this.setLayout(new GridLayout(3,1));
      //将复选框添加到布局管理器
      this.add(jcb);
      this.add(jcb1);
      this.add(jcb2);
      this.add(jcb3);
      this.add(jcb4);
      this.add(jcb5);
      //设置窗口位置和大小
      this.setBounds(10,10,500,500);
      //设置窗口可见
      this.setVisible(true);
      //设置当点击窗口的关闭按钮时退出
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }
    public static void main(String args[])
{
     new Swingfu();
 }
}
```

运行代码，得到结果如图 15-10 所示的结果。



图 15-10　复选框

15.4.6　列表框

列表框是一个十分好用的组件，通过它可以方便收集客户信息。在 Swing 中，用户可以通过 JList 类定义列表框。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/SwingLie.java”：

```
//引入相关包
import java.awt.*;
import javax.swing.*;
public class SwingLie extends JFrame
{
 //定义列表框
    JList jl=new JList(new String[]{"北京","上海","南京","杭州",
    "西安","深圳","重庆"});
    //滚动条
    JScrollPane jsp=new JScrollPane(jl);
    public SwingLie()
 {
        //设置一个 GridLayout 布局管理器
        this.setLayout(new GridLayout(4,3));
        //将滚动条加入布局管理器
        //设置窗口名称
        this.setTitle("列表框");
           this.add(jsp);
        //设置窗口位置和大小
        this.setBounds(10,10,600,200);
        //设置窗口可见
        this.setVisible(true);
        //设置当点击窗口的关闭按钮时退出
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        }
        public static void main(String args[]){
            new SwingLie();
        }
        }
```

运行代码，得到如图 15-11 所示的结果。

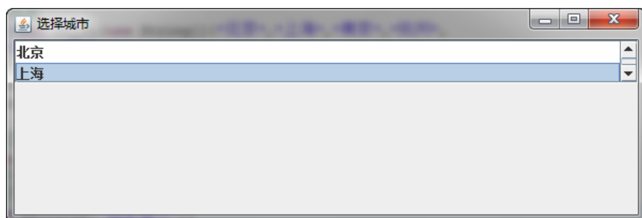


图 15-11　选择城市

15.4.7　选项卡

选项卡是一个十分复杂的组件，它的特点就是多了一个可以放置其他组件的空白区域。在时下相当流行的博客中，就有很多选项卡的应用，通过它们可以把博客变得更有个性。在 Swing 中，用户可以使用 JTabbedPane 类定义选项卡。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/Swingxuan.java”：

```
//引入相关包
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
public class Swingxuan extends JFrame implements ActionListener
{
 //定义选项卡
 JTabbedPane jtb=new JTabbedPane(JTabbedPane.BOTTOM);
 //定义选项
 JPanel xuan1=new JPanel();
 JPanel xuan2=new JPanel();
 JPanel xuan3=new JPanel();
    public Swingxuan()
 {
     //设置窗口名称
     this.setTitle("选择进入的版块");
     //设置一个 GridLayout 布局管理器
     this.setLayout(new GridLayout(1,1));
     //设置选项事件
     xuan1.setBackground(Color.green);
     xuan2.setBackground(Color.red);
     xuan3.setBackground(Color.blue);
```

```
            //将选项卡加入布局管理器
            this.add(jtb);
            //将选项加入选项卡中
            jtb.add("邮箱登录",xuan1);
            jtb.add("进入社区",xuan2);
            jtb.add("进入博客",xuan3);
            //设置窗口位置和大小
            this.setBounds(10,10,580,400);
            //设置窗口可见
            this.setVisible(true);
            //设置当点击窗口的关闭按钮时退出
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
        public void actionPerformed(ActionEvent e)
        {
        }
        public static void main(String args[]){
            new Swingxuan();
        }
        }
```

运行代码，得到如图 15-12 所示的结果。

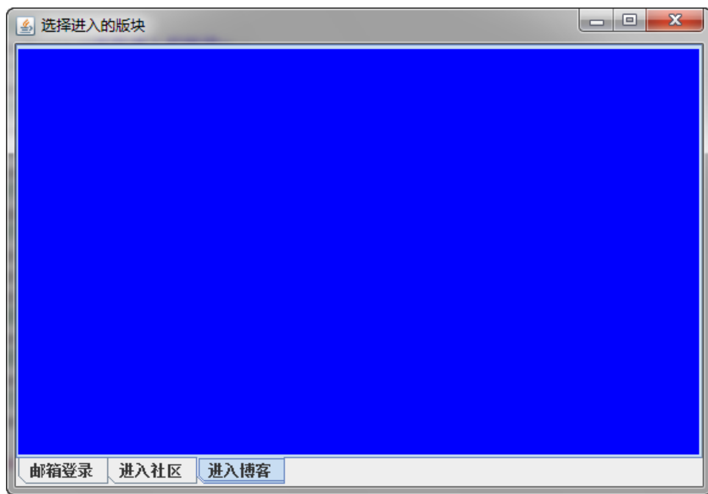


图 15-12 选项卡

15.4.8 文本域

在窗口中，文本框不能接受大量的字符，如果有大量的字符，则需要使用文本域工具。文本域可以接受用户输入多行文本，在 Swing 中一般使用 JtextArea 类定义文本域。其构造方法可以通过查阅 Java API 进行了解。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/SwingAcer.java”：

```
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
public class Swingarea
{
 private JLabel remarkLabel;
 private JTextArea remarkTextArea;
 public void add(JFrame frame)
 {
      frame.setTitle("请输入想说的祝福");
      frame.setLayout(null);
      remarkLabel=new JLabel("祝福语：");
      remarkLabel.setBounds(60, 90, 40, 20);
      remarkTextArea=new JTextArea("在这里输入祝福", 8, 28);
      remarkTextArea.setLineWrap(true);
      JScrollPane   remarkTextAreaScrollPane=new   JScrollPane(remark
TextArea);
      Dimension remarkTextAreaSize=remarkTextArea.getPreferredSize();
      remarkTextAreaScrollPane.setBounds(110,90,remarkTextAreaSize.width,
               remarkTextAreaSize.height);
      frame.add(remarkLabel);
      frame.add(remarkTextAreaScrollPane);
 }
 public static void main(String[] args)
 {
// 创建指定标题的 JFrame 窗口对象
      JFrame frame=new JFrame("利用 JFrame 创建窗口");
// 关闭按钮的动作为退出窗口
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// 设置窗口大小
      frame.setSize(500, 400);
// 获得显示器大小对象
     Dimension  displaySize=Toolkit.getDefaultToolkit().getScreenSize();
;// 获得窗口大小对象
      Dimension frameSize=frame.getSize()
      if (frameSize.width > displaySize.width)
// 窗口的宽度不能大于显示器的宽度
          frameSize.width=displaySize.width;
      if (frameSize.height > displaySize.height)
// 窗口的高度不能大于显示器的高度
          frameSize.height=displaySize.height;
      frame.setLocation((displaySize.width-frameSize.width) / 2,
// 设置窗口居中显示器显示
```

```
                    (displaySize.height-frameSize.height) / 2);
            Swingarea a=new Swingarea();
        // 向 JFrame 窗口添加标签
            a.add(frame);
        // 设置窗口为可见的，默认为不可见
            frame.setVisible(true);
         }
        }
```

运行代码，得到如图 15-13 所示的结果。



图 15-13　执行结果

15.4.9　按钮

按钮通常用来提交组件或者设置其他组件的数据，在 Swing 中，可以使用 JButton 类定义按钮，常用方法有三种，其介绍如下：

- AddActionListener（）：用来单击事件的监听器。
- SetText()：设置按钮文字。
- SetIcon()：设置按钮图标。

下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/SwingButton.java”：

```
        //引入相关包
        import java.awt.*;
        import java.awt.event.*;
        import javax.swing.*;
        public class SwingButton extends JFrame implements ActionListener//继
承 JFrame 类
        {
         //定义一个按钮
         JButton bu=new JButton("你能单击这里吗？");
         public SwingButton()
```

```
        {
            //设置窗口名称
            this.setTitle("按钮");
            //设置一个 GridLayout 布局管理器
            this.setLayout(new GridLayout(5,1));
            //把按钮加到布局管理器中
            this.add(bu);
            //给按钮注册点击事件监听器
            bu.addActionListener(this);
            //按钮设置组件助记符
            bu.setMnemonic('s');
            //按钮设置提示信息
            bu.setToolTipText("单击我");
            //设置窗口位置和大小
            this.setBounds(10,10,450,400);
            //设置窗口可见
            this.setVisible(true);
            //设置当点击窗口的关闭按钮时退出
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
        //实现 ActionListener 类中的 actionPerformed(ActionEvent e)抽象方法
        public void actionPerformed(ActionEvent e)
        {
            this.bu.setText("谢谢单击");
        }
        public static void main(String args[])
        {
            new SwingButton();
        }
       }
```

运行代码，得到如图 15-14 所示的结果。



图 15-14　按钮

15.4.10　进度条

进度条在局域网领域使用得比较少，但在广域网中应用得较多，如下载常常需要使用进度条。Swing 里也有进度组件，用户可以通过以下方法设置进度条：

- setMaximun(int n)：设置最大进度值。
- setMinimun(int n)：设置最小进度值。
- setString(String s)：设置在进度条上显示的文字内容。
- setStringOaubted(Boolean b)：设置在进度条上绘制文字。
- setIndeteminate(Boolean new value)：设置进度条的显示模式。
- getValue()：获取当前进度值。
- setValue(String s)：设置当前进度值。

下面通过一段代码讲解进度条，其代码见“光盘：源代码/第 15 章/Swing jindu.java”：

```
//引入相关包
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;
import javax.swing.event.*;

public class Swingjindu extends JFrame
{
//定义进度条
    JProgressBar xia=new JProgressBar();
    public Swingjindu()
 {
     //设置窗口名称
     this.setTitle("正在下载");
     //设置一个 GridLayout 布局管理器
     this.setLayout(new GridLayout(4,1));
     //进度条加入到布局管理器
     this.add(xia);
     //设置进度条的最大刻度值
        xia.setMaximum(100);
        //设置进度条的最小刻度值
     xia.setMinimum(0);
     //设置当前进度值
     xia.setValue(20);
     //设置在进度条上显示的文字内容
     xia.setString(35+"%");
     //设置是否在进度条上绘制文字
     xia.setStringPainted(true);
     //设置进度条为动态显示模式
     xia.setIndeterminate(true);
```

```
        //设置窗口位置和大小
        this.setBounds(10,10,620,480);
        //设置窗口可见
        this.setVisible(true);
        //设置当点击窗口的关闭按钮时退出
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String args[])
    {
        new Swingjindu();
    }
}
```

运行代码，得到如图 15-15 所示的结果。

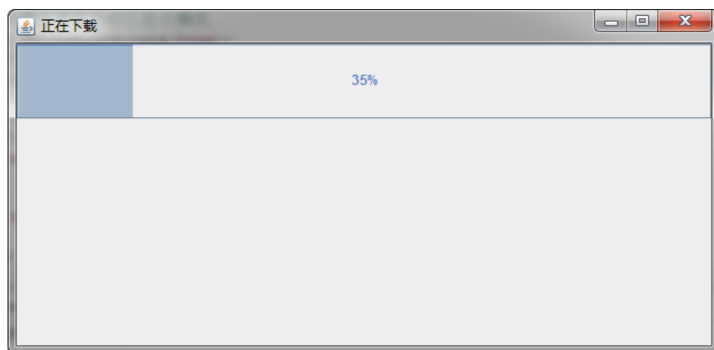


图 15-15　进度条

15.5　常用的布局管理器

Swing 窗口在默认情况下的布局方式比较单一，可能满足不了开发者的需求。但 Swing 窗口为满足不同开发者的需求，提供了多种布局方式，在本节中，将详细讲解开发者经常使用的 Swing 窗口布局管理器、边界布局管理器、网格布局管理器或者不使用布局管理器。

15.5.1　不使用布局管理器

在布局管理器出现之前，组件都是使用直接定位的方法排列在容器中的，如 VC、VB 等开发语言都使用这样的方式排列容器，Java 也提供了针对绝对定位的组件排列方式的支持。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/SwingJuedui.java”：

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
```

```
import javax.swing.WindowConstants;
public class SwingJuedui extends JFrame
{
 private JLabel userLabel;
 private JLabel passLabel;
 private JButton exit;
 private JButton login;
 private JTextField userName;
 private JPasswordField userPassword;
 public SwingJuedui()
 {
      setTitle("登录窗口");
      setBounds(300, 200, 400, 350);
      getContentPane().setLayout(null);
      userLabel=new JLabel();
      userLabel.setText("用户名：");
      userLabel.setBounds(10, 10, 200, 18);
      getContentPane().add(userLabel);
      userName=new JTextField();
      userName.setBounds(60, 10, 200, 18);
      getContentPane().add(userName);
      passLabel=new JLabel();
      passLabel.setText("密   码：");
      passLabel.setBounds(10, 50, 200, 18);
      getContentPane().add(passLabel);
      userPassword=new JPasswordField();
      userPassword.setBounds(60, 50, 200, 18);
      getContentPane().add(userPassword);
      login=new JButton();
      login.setText("进入");
      login.setBounds(90, 80, 60, 18);
      getContentPane().add(login);
      exit=new JButton();
      exit.setText("退出");
      exit.setBounds(170, 80, 60, 18);
      getContentPane().add(exit);
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
 }
 public static void main(String[] args)
 {
      SwingJuedui login=new SwingJuedui();
      login.setVisible(true);
 }
}
```

运行代码，得到如图 15-16 所示的结果。



图 15-16　绝对定位组件

提示：这样布局的最大的缺点是在不同的操作系统下，显示情况不同，如 Windows xp 和 Windows Vista 和 Linux 下，显示的结果可能是各种各样的。

15.5.2　使用边界布局管理器

边界布局管理器实际上在学习 AWT 布局时便提到过，就是东西南北中五个位置的布局方式。Swing 中的布局管理器，与 AWT 稍有不同。下面通过一段代码讲解边界布局管理器布局，其代码见“光盘：源代码/第 15 章/Swingbian.java”：

```
import java.awt.BorderLayout;
import java.awt.Container;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.WindowConstants;
public class Swingbian extends JFrame
{
 private JButton northButton;
 private JButton southButton;
 private JButton centerButton;
 private JButton westButton;
 private JButton eastButton;
 public Swingbian()
 {
      setTitle("神奇的边界布局");
      setBounds(100, 100, 500, 400);
      northButton=new JButton("北丐—洪七公");
      southButton=new JButton("南帝-段智兴");
      westButton=new JButton("西毒-欧阳锋");
      eastButton=new JButton("东邪-黄药师");
      centerButton=new JButton("中神通-王重阳");
      Container panel=getContentPane();
      panel.setLayout(new BorderLayout());
      panel.add(northButton, BorderLayout.NORTH);
      panel.add(southButton, BorderLayout.SOUTH);
      panel.add(westButton, BorderLayout.WEST);
      panel.add(eastButton, BorderLayout.EAST);
      panel.add(centerButton, BorderLayout.CENTER);
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
 }
 public static void main(String[] args)
```

```
        {
            Swingbian demo=new Swingbian();
            demo.setVisible(true);
        }
       }
```

运行代码，得到如图 15-17 所示的结果。

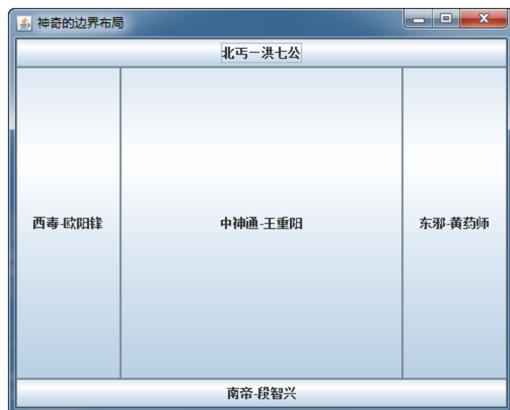


图 15-17　边界布局管理器

15.5.3　使用网格布局管理器

网格布局管理器在讲解 AWT 时便有所提及，就是在第 14 章中使用 Gridlayout 布局的内容。如图 15-18 所示。



图 15-18　网格布局

在 Swing 中自然也可以使用网格布局管理器，且它的功能更为强大。下面通过一段代码进行讲解，其代码见“光盘：源代码/第 15 章/Swingwang.java”：

```
       import java.awt.Container;
       import java.awt.GridLayout;
```

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.WindowConstants;
public class Swingwang extends JFrame
 {
 private JButton[] buttons;
 public Swingwang()
 {
      setTitle("边界布局管理器实例");
      setBounds(100, 100, 500, 400);
      buttons=new JButton[14];
      Container panel=getContentPane();
      panel.setLayout(new GridLayout(0,3));
      for(int i=0;i<13;i++)
{
           buttons[i]=new JButton("按钮"+(i+1));
           panel.add(buttons[i]);
      }
      setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
 }
 public static void main(String[] args)
 {
      Swingwang demo=new Swingwang();
      demo.setVisible(true);
 }
}
```

运行代码，得到如图 15-19 所示的结果。



图 15-19　网格布局

15.6　疑难问题解析

本章详细讲解了 Swing 创建窗口、添加组件与常用的 Swing 窗口布局等知识。本节将对

本章中较难理解的问题进行讲解。

读者疑问：在使用 Swing 创建的窗口中，使用 Icon 接口显示图片，图片的位置放在何处？为什么在执行源程序时，却没有发现图片？

解答：Swing 图片位置是一个相对路径，放在相关目录下就可以了。如果是使用 Eclipse 开发项目，用户只需将图片放在项目的根目录下就可以了。如果是直接使用“DOS”命令开发和调试 Java 源程序，用户需要将图片和源文件放置在一起。

读者疑问：在开发 Swing 窗口的过程中，想开发某一功能，却不知道方法怎么办？

解答：如果用户需要开发某一功能却不知道方法，可以查询 Java API 手册，在 Java API 手册里可以获得相应的方法，去实现相应的功能。



职场点拨——你准备找好工作吗

在职场中的 IT 精英们肯定会有寻找更好的工作的想法，但是换工作是既有利也有弊的。企业培养一个程序员不容易，辛苦地培养起来一个人，熟悉了业务，掌握了技术，这时候走人，损失最大的当然是企业，花时间培养人和熟悉业务也是需要成本的。对程序员来说，跳槽几乎是利大于弊，首先待遇上肯定会水涨船高，不然就是跳槽失败，除非有其他想法和目标。

换工作对于技术人员来说并不是坏事，主要有如下两方面原因。

1）一般的 IT 公司都有自己常用的模式，该模式经过一个项目的打磨之后，就可以基本掌握，相关知识、架构等大概也可以了解，此时可以换个环境寻找更高的发展。

2）跳槽相当于变向升职，这个可以从你的简历中体现出来。

在换工作时，是否能够获得满意的岗位是需要技巧的。不同的级别会有不同的技巧，接下来将一一讲解。

（1）初级程序员

当做完一个项目的时候，你会了解这个项目的整个流程，此时可以在简历中填写中级程序员的角色，把很多中级程序员做的事情写到你的履历里（前提是你要了解这些），跳槽的时候，你的目标自然就是中级程序员，而招聘公司看到你的情况也会觉得合适。

（2）中级程序员

需要在项目中了解高级程序员的工作范围，并不要求你全部掌握，但需要你能表达出来，这个很重要。比如后台的设计模式、软件架构、接口设计等，把这些写到你的履历中，给自己定位成高级程序员，自然的，需要高级程序员的公司会找到你。

（3）高级程序员

你所需要了解的就不仅仅是程序设计，而是整个项目的运作和管理流程。包括项目管理、系统架构（软硬件）、系统集成等，整个环节不一定都要会，但需要知道是什么，比如，什么是交换机，什么是硬件负载均衡设备，什么是反向代理，什么是缓存服务器，什么是 WEB 服务器，什么是集群、负载均衡、分布式、数据库优化、大数据存储、高并发访问等等，都是你需要了解的，面试的时候能表达出来，那么你就成功了。同样可以把这些写到你的履历中，给自己定位架构师或项目经理，更新简历后，猎头会来找你。

（4）系统架构师

既然选择了架构师的角色，那么肯定是向技术方向发展了。技术总监、研发总监甚至CTO 就是你的目标。技术总监需要负责整个公司的技术部运作，包括对人员的管理、绩效考核、各语言组之间的协调、各项目间的协调，各部门间的协调，除此之外，你还需要考虑所运营的项目如何发展得更好，网站如何才能更加优化，产品如何能更上一个层次，公司的技术发展如何规划，各种方案如何快速编写和实施，如何与老板打交道等，都是你需要掌握的。

（5）项目经理

项目经理分两种，一种是 TEAM LEADER 的角色，需要很强的技术；一种是负责招标、流程控制的偏商务角色，要懂技术。发展到这个层次的，我想应该不用我来告诉他们出路在哪里了，这样的人一般都有自己的规划，但凡事都有例外，如果没有规划或发展迷茫的，TEAM LEADER 角色可以重点把项目管理、人力资源、系统架构等环节再强化一下，紧跟当前发展形势学习新知识；而偏商务角色的，则可以考虑向总经理、CIO、CEO 等方向努力，到这个层次的，需要的不仅仅是知识，更多的是一种理念和个人魅力。

第 16 章　Java 和数据库

数据库是用来存储数据的仓库，它在软件开发中的作用不言而喻。现如今的数据库软件很多，如 MySQL、DB2 和 SQL Sever 等等，本书主要讲解 MySQL，初学者首先应该熟练使用 MySQL，然后再学习数据库的知识，只有这样，才能真正领会数据库的精髓。本章主要内容如下：

- 数据库的定义。
- MySQL 的安装。
- MySQL 的管理工具。
- SQL Sever 数据库。
- 职场点拨——我有一颗创业的心。

2010 年 XX 月 XX 日，小雪

我今天得知大学同学 B 创业有成，不但买了车、买了房，就在昨天，还举办了一场风光无限的婚礼。我很羡慕他，我也想通过创业来改变自己的命运！



一问一答

小菜：“我想自己创业，不知是否可行？”

Wisdom：“很不错，原来想跳槽，现在想创业，都是积极向上的想法。创业成功会使你得到想要的一些东西，但是万一失败，你也会失去很多东西，还会对你的信心造成巨大的打击。所以你得三思而后行，建议你先读读本章最后的‘我有一颗创业心’内容，也许能给你带来一些启发。”

小菜：“言归正传，数据库很重要吗？”

Wisdom：“你知道是什么改变了软件行业的命运呢？就是数据库，数据库是实现动态应用的最佳工具，是当今大型应用程序开发必不可少的组成部分。其中和 Java 结合最好的数据库是 MySQL，当安装好 MySQL 数据库软件和 MySQL 管理软件后，用户就可以根据需要随心所欲地创建数据库和表。”

16.1　数据库的定义

数据库通常分为层次式数据库、网络式数据库和关系式数据库三种。不同的数据库是按不同的数据结构来联系和组织的，数据库存储着许多数据，这些数据具备以下特点：

- ❑ 实现数据共享：数据共享包含所有用户可同时存取数据库中的数据，也包括用户可以用各种方式通过接口使用数据库，并提供数据共享。
- ❑ 减少数据的冗余度：同文件系统相比，由于数据库实现了数据共享，从而避免了用户各自建立应用文件。减少了大量重复数据，减少了数据冗余，维护了数据的一致性。
- ❑ 数据的独立性：数据的独立性包括数据库中数据库的逻辑结构和应用程序相互独立，也包括数据物理结构的变化不影响数据的逻辑结构。
- ❑ 数据实现集中控制：文件管理方式中，数据处于一种分散的状态，不同的用户或同一用户在不同处理中其文件之间毫无关系。利用数据库可对数据进行集中控制和管理，并通过数据模型表示各种数据的组织以及数据间的联系。
- ❑ 数据一致性和可维护性，以确保数据的安全性和可靠性，主要包括：①安全性控制：以防止数据丢失、错误更新和越权使用；②完整性控制：保证数据的正确性、有效性和相容性；③并发控制：使在同一时间周期内，允许对数据实现多路存取，又能防止用户之间的不正常交互作用；④故障的发现和恢复：由数据库管理系统提供一套方法，可及时发现故障和修复故障，从而防止数据被破坏。

16.2 操作 MySQL 数据库

操作 MySQL 数据库的步骤如下：

1）启动 MySQL 管理软件，单击“连接”按钮，在弹出的窗口中输入数据库的相关信息，单击“确定”按钮，如图 16−1 所示。

2）打开管理软件后，用户可以看到左侧列出的数据库是灰色的，这表示它们是不能使用的，如果需要使用，需要选中这个数据库，单击鼠标右键，在弹出的快捷菜单中选择“打开数据库”命令。这里新建一个数据库，选择“localhost”单击鼠标右键，在弹出的快捷菜单中选择“创建数据库”命令，如图 16−2 所示。

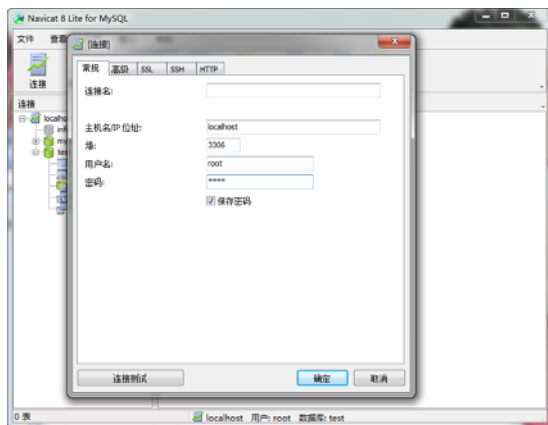


图 16−1 “连接”对话框

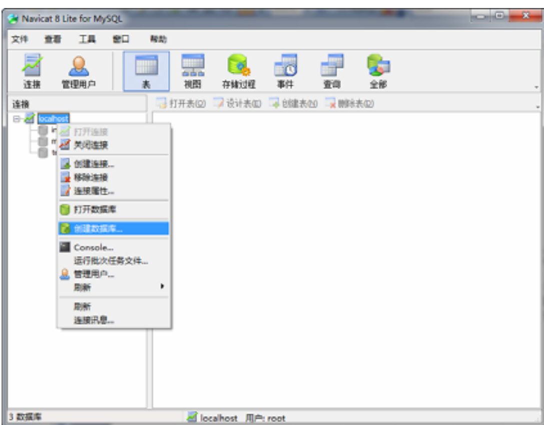


图 16−2 选择“创建数据库”命令

3）打开“创建新数据库”窗口，在“键入数据库名”文本框中输入名称，这里输入“book”，然后在“字符集”下拉列表中选择“utf8”选项，单击“确定”按钮，如图 16−3 所示。

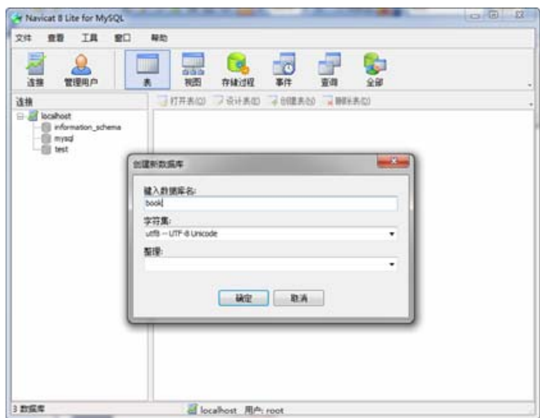


图 16-3　“创建新数据库”窗口

4）选中新建的数据库，单击鼠标右键，在弹出的快捷菜单中选择“打开数据库”命令，然后再选中表，单击鼠标右键，选择“创建表”命令，如图 16-4 所示。

5）打开“表设计器”，设置一张表的字段，如图 16-5 所示。

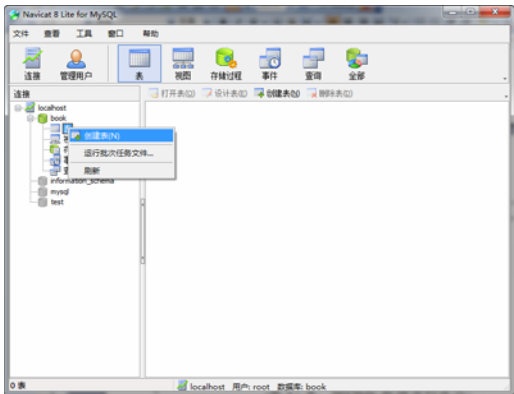


图 16-4　选择“创建表”命令

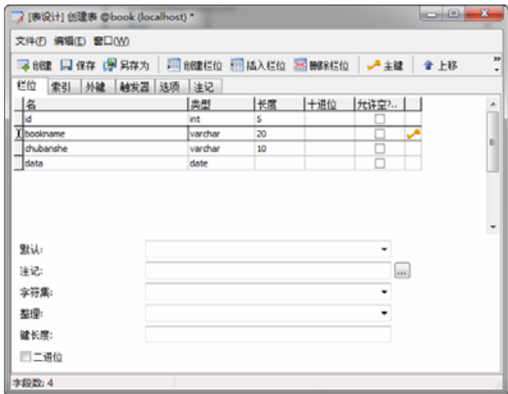


图 16-5　设置字段

6）字段设置完成后，单击“表设计器”里的“保存”按钮，会弹出“表名”窗口，在“键入表名”文本框中输入“book”，单击“确定”按钮，如图 16-6 所示。

7）回到窗口主界面，单击“book”表，用户可以根据需要创建记录，如图 16-7 所示。

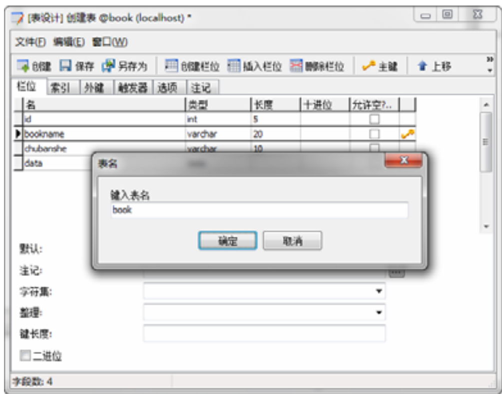


图 16-6　输入表名

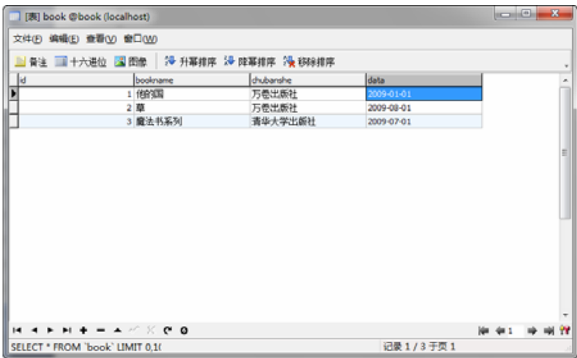


图 16-7　创建记录

16.3 MySQL 的安装

MySQL 软件的安装向导是全英文的界面，需要设置的东西较多，本节将详细讲解 MySQL 的设置和安装步骤，其具体操作如下：

1）双击 MySQL 的安装程序，打开如图 16-8 所示的窗口，单击“Next”按钮。

2）在打开的“Setup Type”窗口中有三个选项，第一个选项是“典型安装”，第二个是“完全安装”，第三个是“自定义安装”，这里选择第三个选项，然后单击“Next”按钮，如图 16-9 所示。

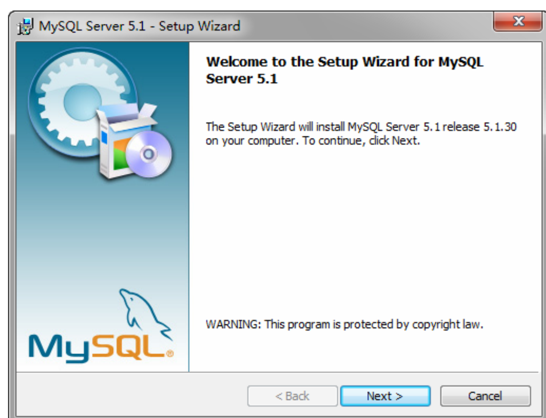


图 16-8　安装向导

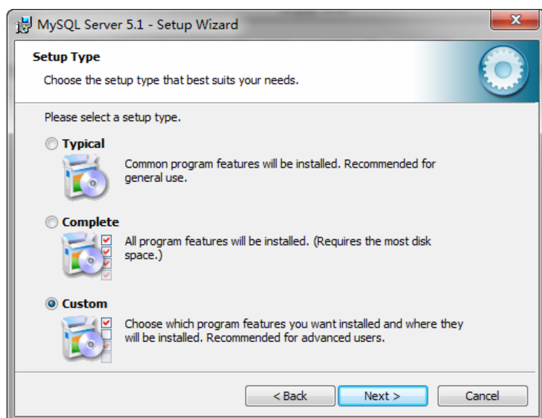


图 16-9　选择需要的版本

3）打开“Custom Setup”窗口，设置安装路径，这里保持默认设置，单击“Next”按钮，如图 16-10 所示。

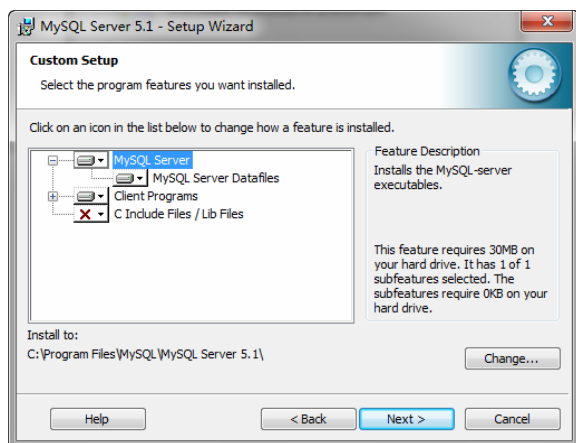


图 16-10　安装位置

4）在打开的窗口中单击“Install”按钮，等待片刻，将会进行安装，如图 16-11 所示。

5）安装完成后会弹出一个对话框，单击“Next”按钮，在弹出的窗口中继续单击“Next”按钮，如图 16–12 所示。

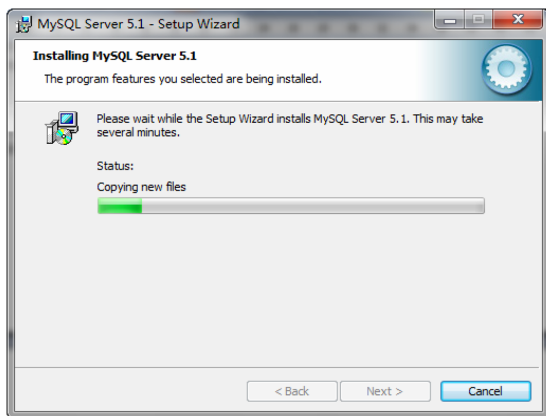


图 16–11　安装过程

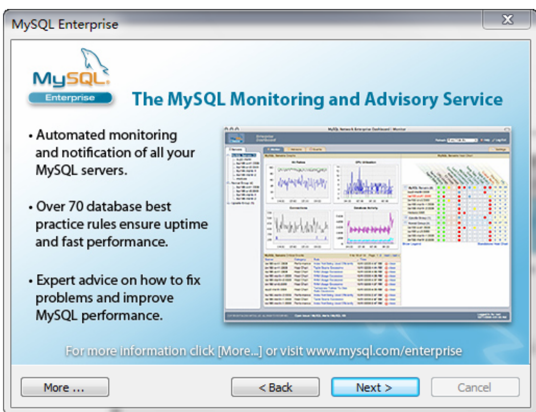


图 16–12　安装

6）等待片刻，将会打开如图 16–13 所示的窗口，单击“Finish”按钮。

7）弹出如图 16–14 所示的窗口，单击“Next”按钮。

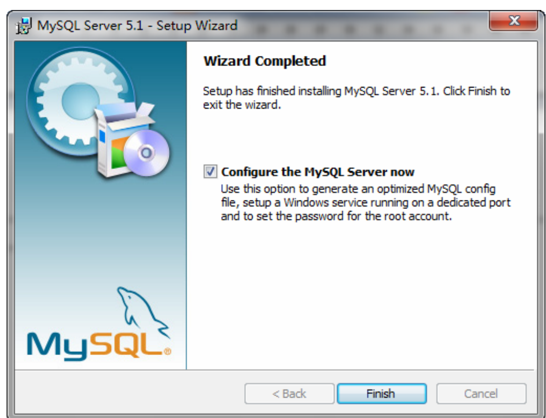


图 16–13　完成安装

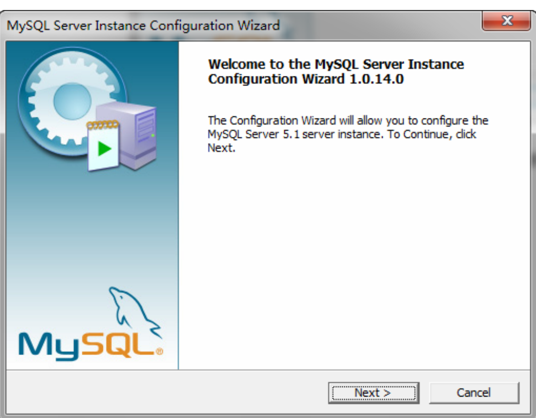


图 16–14　设置第一个步骤

8）在打开的窗口中有两个选项，第一个是详细设置，第二个是标准设置，选择第一个选项，单击“Next”按钮，如图 16–15 所示。

9）在打开的窗口中，有三个版本可供用户选择，第一个是开发者版本，第二个是服务器版本，第三个是专业的 MySQL 使用者版本，这里选择默认的“开发者版本”进行安装，单击“Next”按钮，如图 16–16 所示。

10）在打开的窗口中单击“Next”按钮，然后在弹出的窗口中选择数据的存放位置，这里使用默认设置，单击“Next”按钮，如图 16–17 所示。

11）在打开的窗口中单击“Next”按钮，在弹出的窗口中设置端口号等信息，使用默认

设置即可，单击“Next”按钮，如图 16-18 所示。

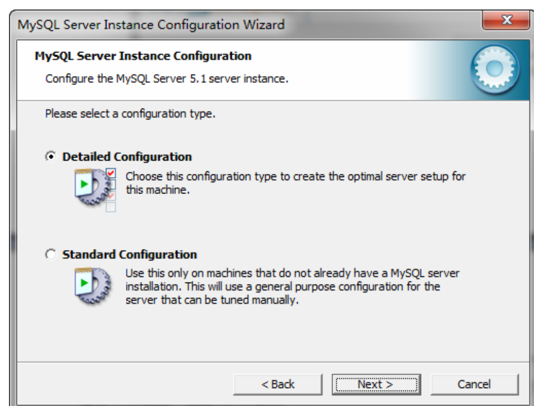


图 16-15　完全设置

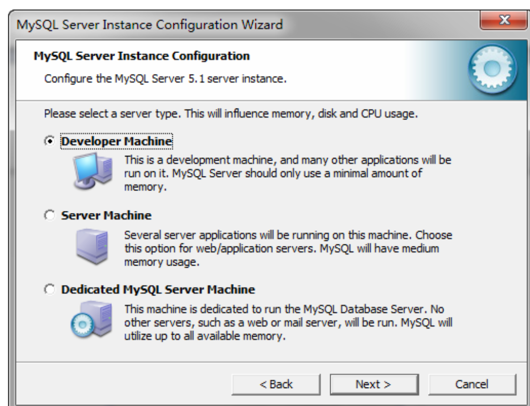


图 16-16　选择需要的版本

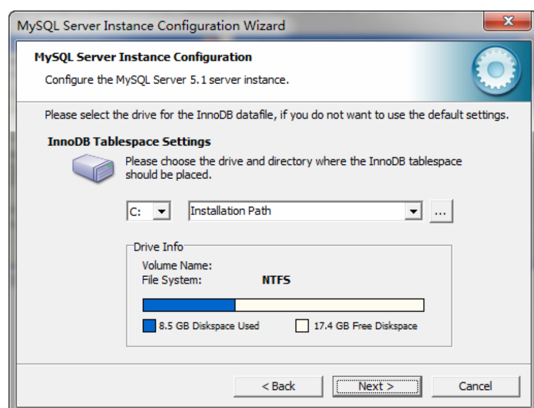


图 16-17　设置路径

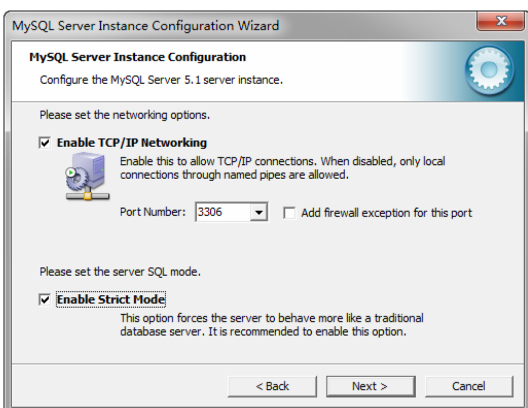


图 16-18　设置 MySQL 服务器

12）在打开的窗口中选择第三个选项，然后在“Character set”下拉列表中选择“utf8”，单击“Next”按钮，如图 16-19 所示。

13）在打开的窗口中设置密码，单击“Next”按钮，如图 16-20 所示。

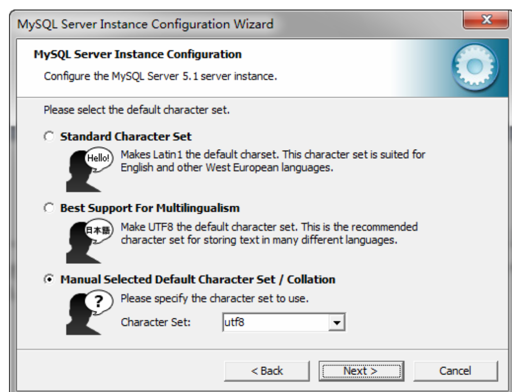


图 16-19　设置语言选项

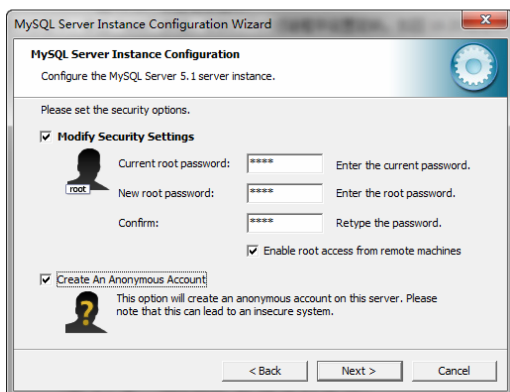


图 16-20　设置密码

14）在打开的窗口中单击“execute”按钮，将执行命令。完成后单击“Finish”按钮，如图 16−21 所示。

15）启动 MySQL Command Line Client 程序，输入密码后进入 MySQL 管理，使用 SQL 语句管理数据库，如图 16−22 所示。

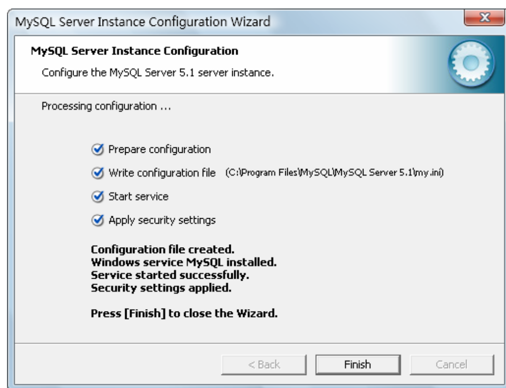


图 16−21　安装完成

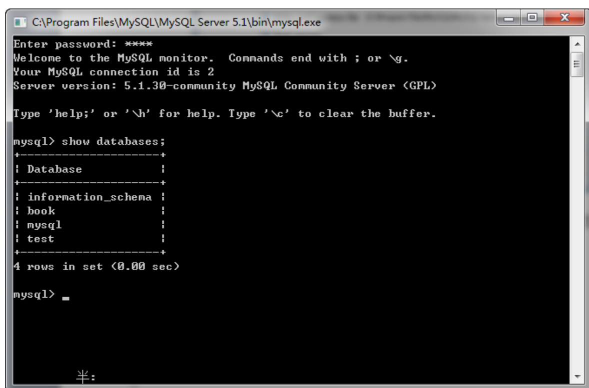


图 16−22　管理工具

16.4　MySQL 的管理工具

MySQL 的管理工具很多，如 PHPmyadmin、SQLyog 和Navicat等，这里只讲解Navicat 管理工具，用户可以去网站免费获取，下载后进行安装，安装完成后，即可对 MySQL 进行管理。

16.4.1　创建数据库

准备好 MySQL 数据库和管理工具，接下来就该创建数据库了。创建数据库十分简单，其具体操作如下：

1）启动Navicat Lite for MySQL 管理软件，在窗口左侧选中 localhost，单击鼠标右键，在弹出的快捷菜单中选择“新建数据库”命令，如图 16−23 所示。

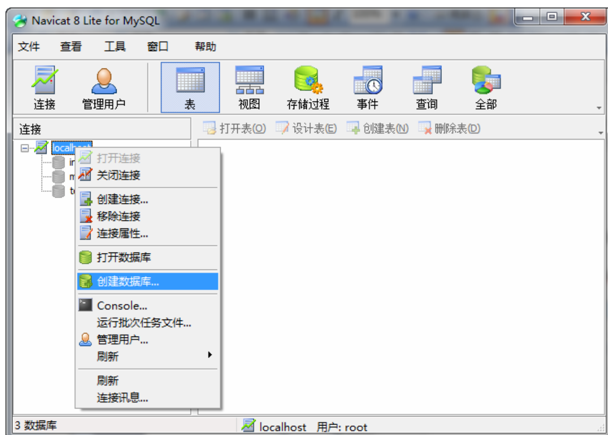


图 16−23　选择命令

2）打开“创建数据库”窗口，在“键入数据库名称”文本框中输入“news”，单击“确定”按钮，如图 16–24 所示。

提示：创建好数据库后，用户一定要正确选择字符集，在安装 MySQL 时设置了“utf8”，这里仍然需要设置，否则将会影响显示。

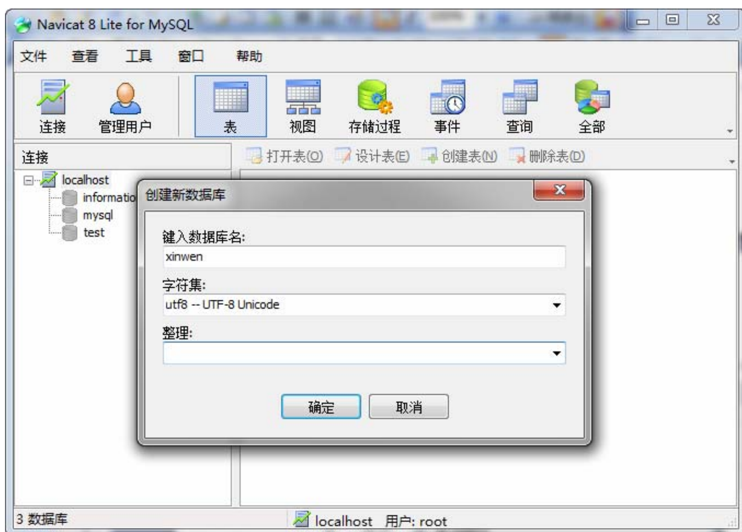


图 16–24 “创建新数据库”对话框

16.4.2 创建表

一个数据库中有多个表，创建数据库表的方法十分简单，其具体操作如下：

1）单击窗口左侧数据库下的表，选中“表”选项，单击鼠标右键，选择“创建表”命令，如图 16–25 所示。

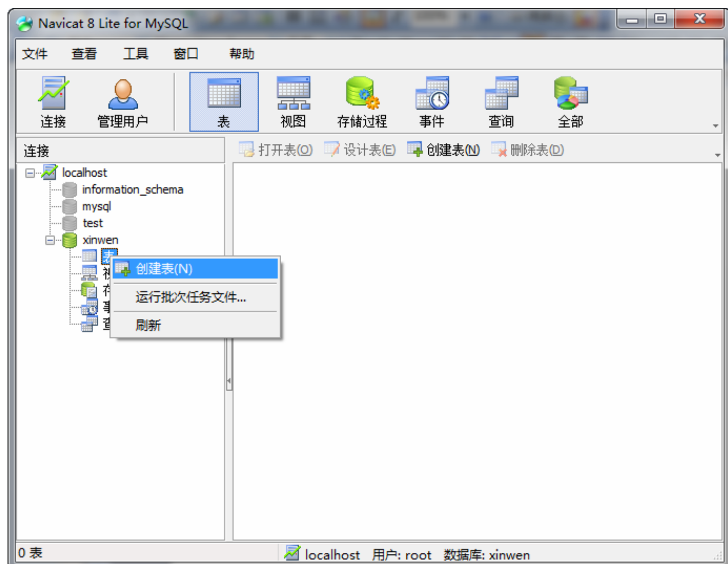


图 16–25 选择命令

2）打开表设计器，在“名”中输入字段，并设置字段类型和字段长度，如图 16–26 所示。

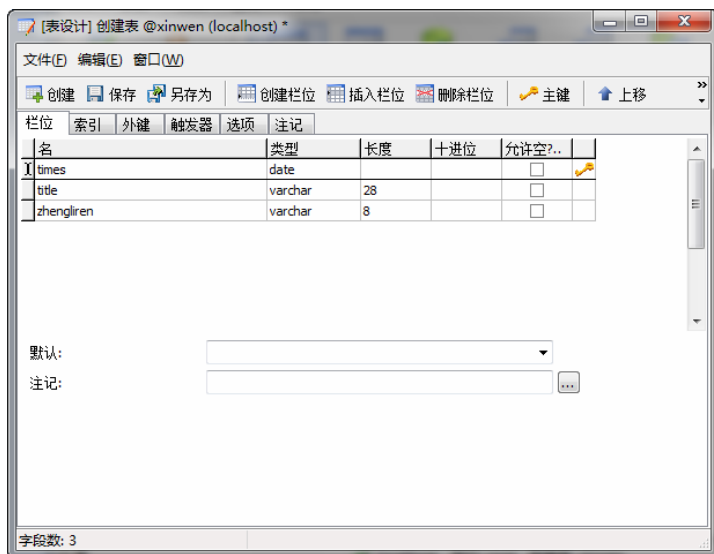


图 16–26　创建字段

3）设置完成后，单击“保存”按钮，在弹出的“表名”窗口的文本框中输入表名，这里输入“news”，之后单击“确定”按钮，如图 16–27 所示。

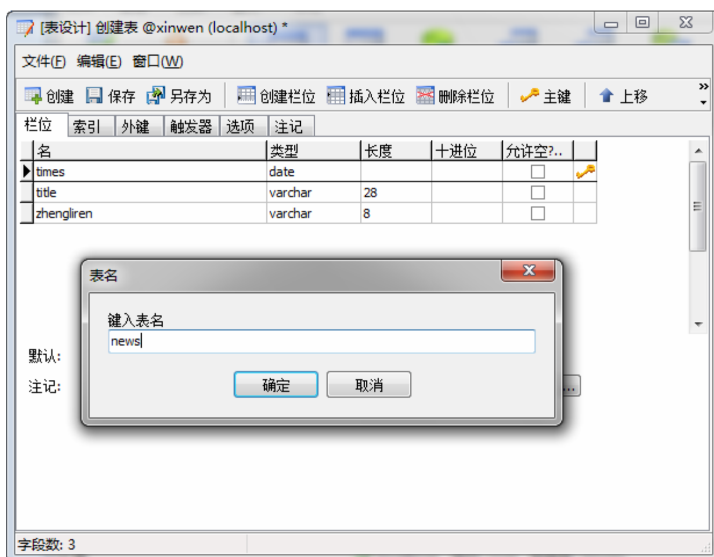


图 16–27　输入表名

16.4.3　输入记录

记录实际上就是信息，它是数据库表中一行的称呼。记录对于数据库来说十分重要，下面讲解其具体操作步骤：

1）选中需要创建记录的表，单击“打开表”按钮，如图 16-28 所示。

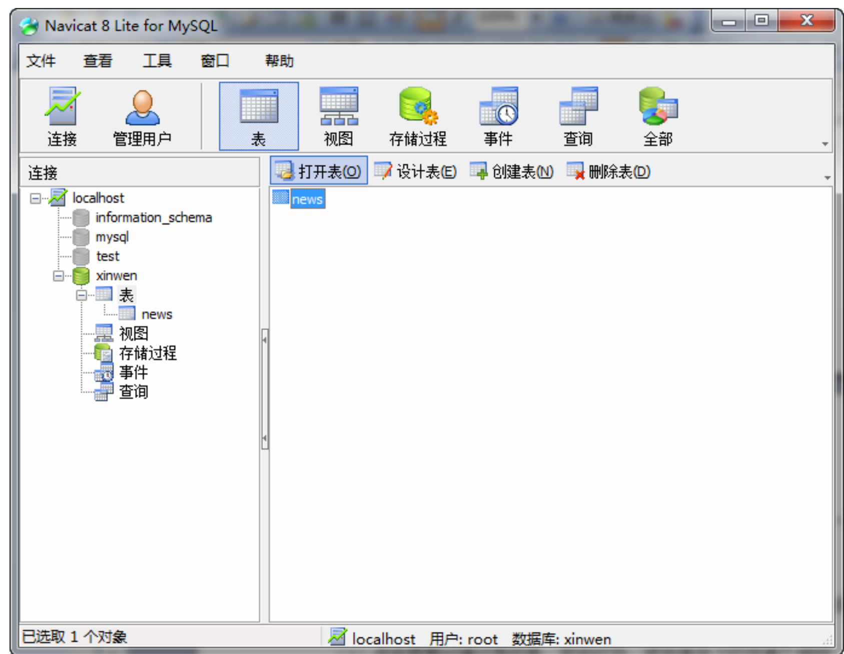


图 16-28　打开表

2）在设置字段输入对应的值，如果需要插入一条新的记录，单击窗口左下角的“+”按钮，如图 16-29 所示。

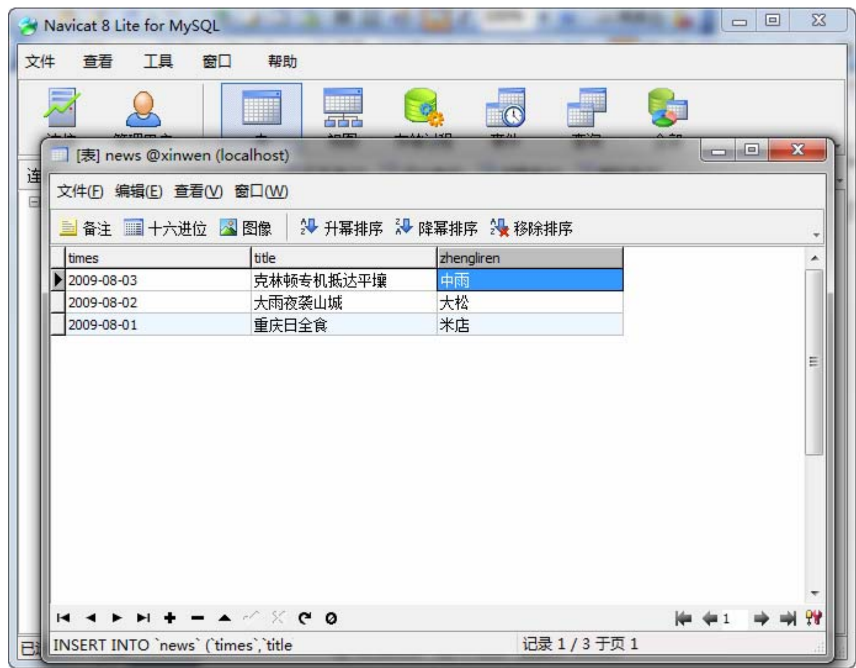


图 16-29　输入记录

16.5　SQL Sever 很简单

Microsoft 公司有一款十分优秀的网络数据库，那就是 SQL Sever，其功能非常强大，且应用广泛，但是最适用的是 SQL Sever 2000。本节将讲解 SQL Sever 2000。

16.5.1　创建数据库

1）完成 SQL Sever 2000 的安装后，单击“开始”菜单，选择“所有程序/Microsoft SQL Sever/企业管理器”命令，启动 SQL Sever 2000，如图 16−30 所示。



图 16−30　选择命令启动 SQL Sever 2000

2）在窗口左侧的树形列表中选中“数据库”选项，然后单击鼠标右键，选择“新建数据库”命令，如图 16−31 所示。

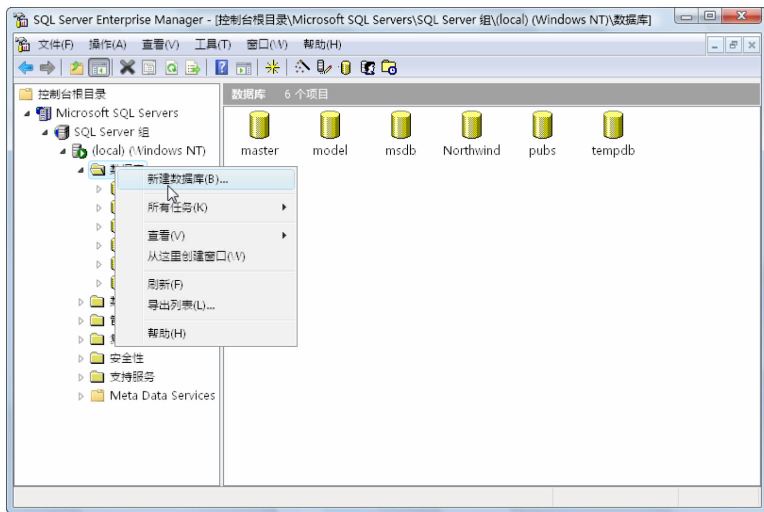


图 16−31　新建数据库

3）打开“数据库属性”窗口，在“名称”文本框中输入“news”，如图 16−32 所示。

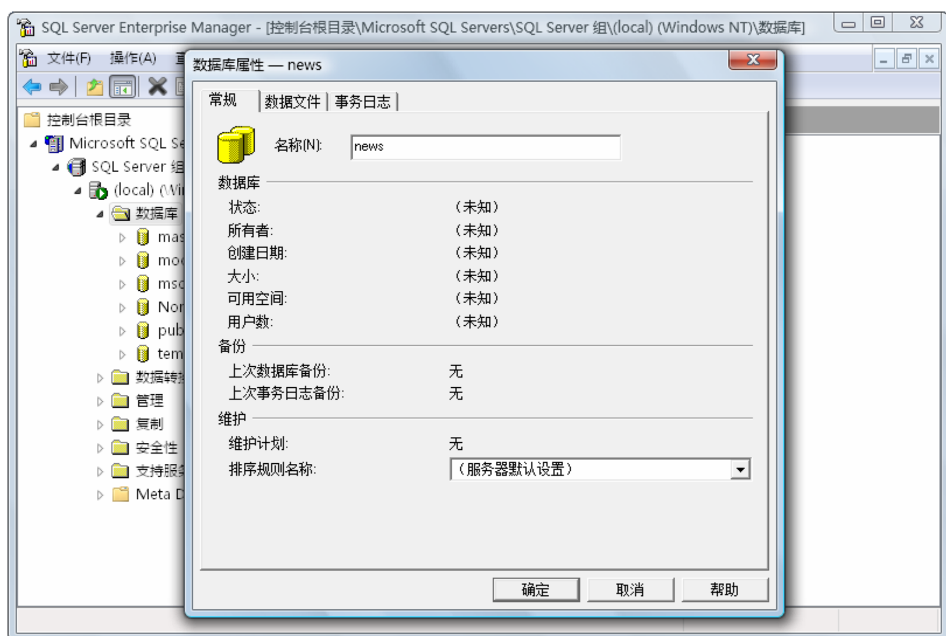


图 16−32　输入数据库名称

4）打开“数据文件”选项卡，设置数据库文件的位置（默认情况下，数据库是放在系统盘下的），用户可以单击按钮，打开“查找数据库文件”窗口，根据需要指定一个位置，这里指定的是 F 盘，如图 16−33 所示。

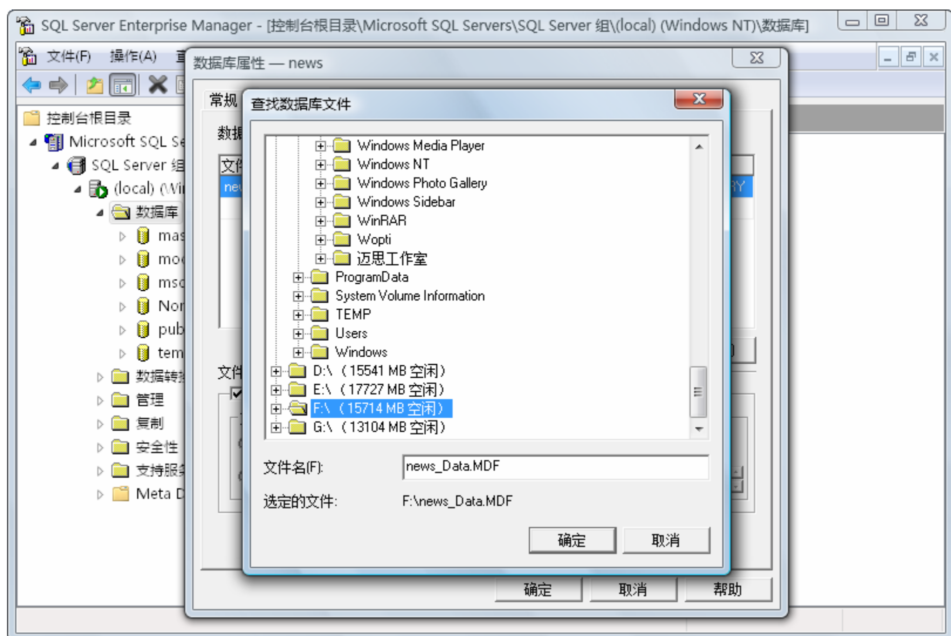


图 16−33　更改数据库文件

5）单击打开“事务日志”选项卡，设置数据库文件的位置（默认情况下，数据库是放在系统盘下的），用户可以单击▭按钮，打开“查找事务日志文件”窗口，根据需要指定一个位置，这里指定的是 F 盘，如图 16–34 所示。

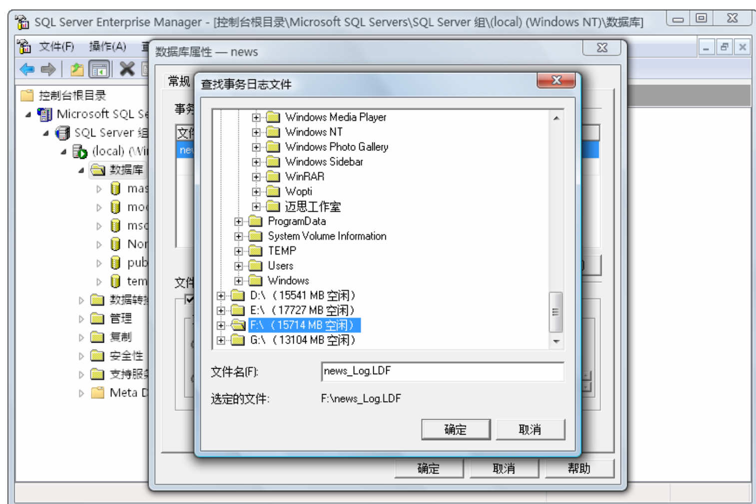


图 16–34　指定事务日志文件的位置

6）设置完成后，依次单击“确定”按钮即可。

16.5.2　创建表

创建好数据库后，用户可以继续创建数据库中的表，其具体操作步骤如下：

1）单击需要创建表的数据库，在属性列表中选中“表”选项，单击鼠标右键，选择“新建表”命令，如图 16–35 所示。

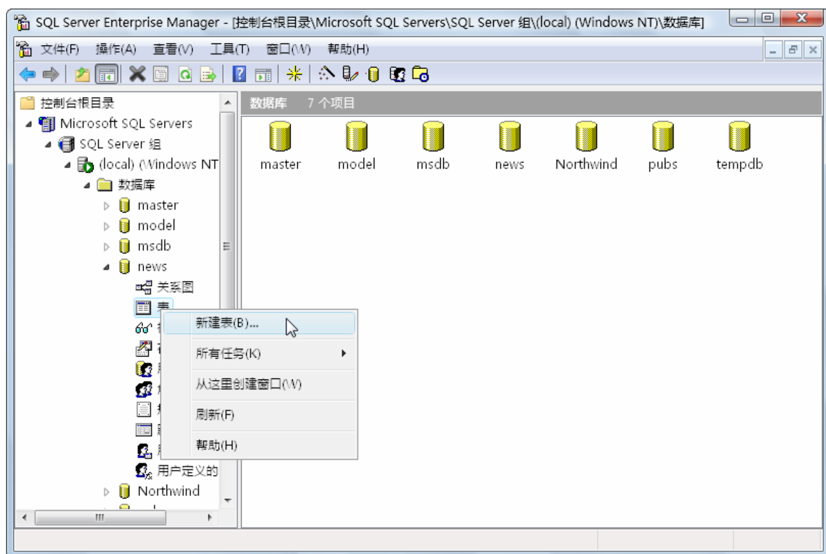


图 16–35　选择“新建”命令

2）在打开的窗口中，需要为这个表创建字段，然后为字段设置属性，如图 16-36 所示。

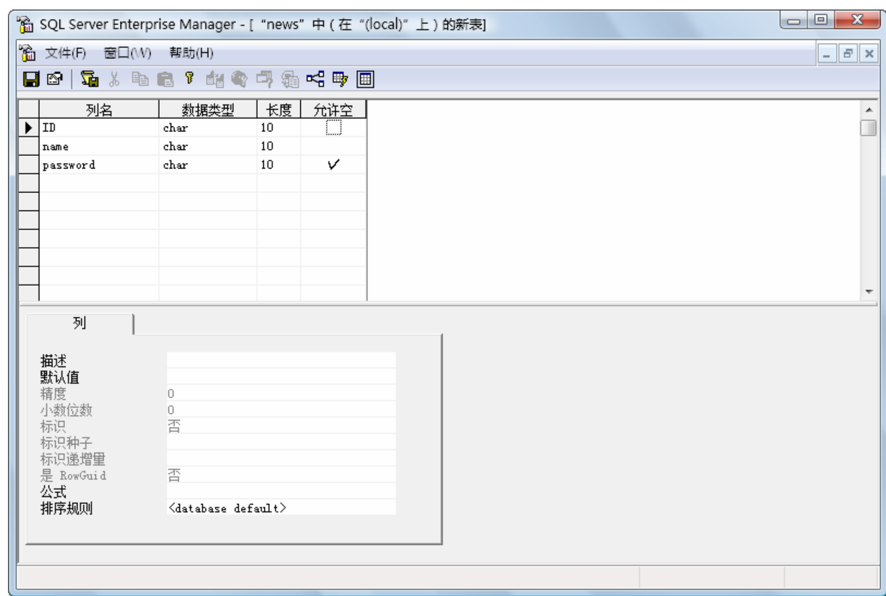


图 16-36　设置字段

3）设置完成后，单击“保存”按钮，打开“选择名称”窗口，在“输入表名”文本框中输入名称，这里输入“user”，单击“确定”按钮，如图 16-37 所示。

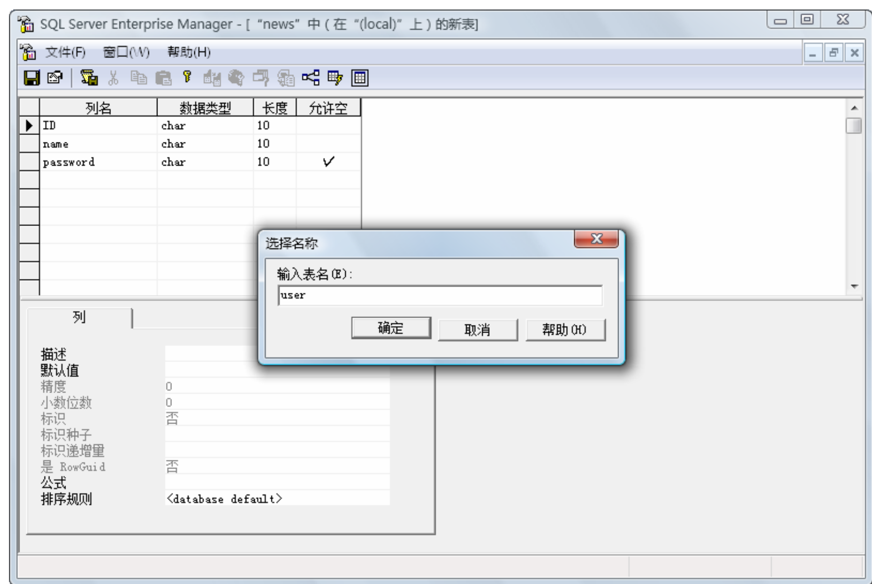


图 16-37　输入表名

4）回到数据库的窗口，查看新建的表，如图 16-38 所示。

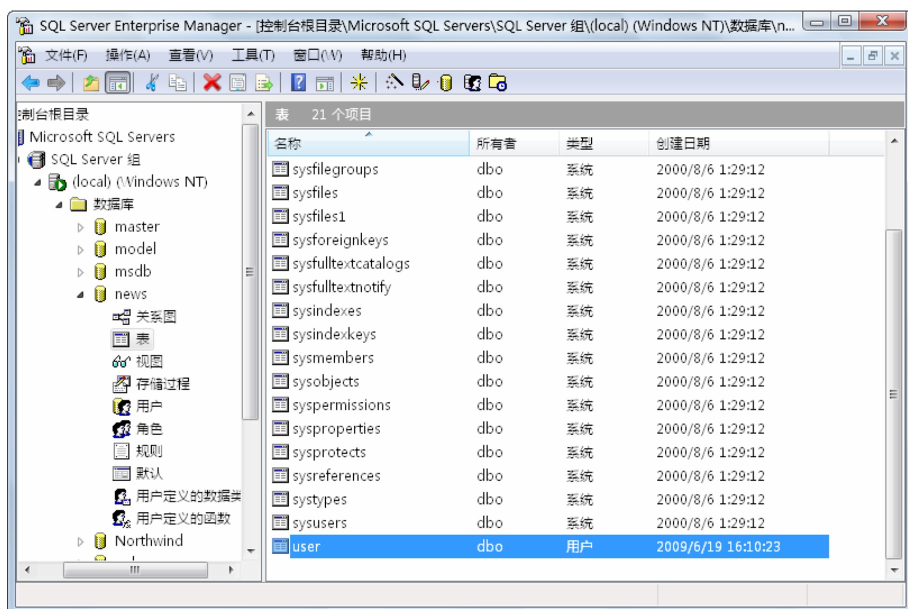


图 16-38　新建表

16.5.3　创建记录

当表创建完成后，接下来要做的就是输入记录，输入记录的过程和 MySQL、Access 数据库十分类似，其具体操作如下：

1）选中需要创建记录的表，如“user”表，然后单击鼠标右键，在弹出的快捷菜单中选择“打开表/返回所有行”命令，如图 16-39 所示。

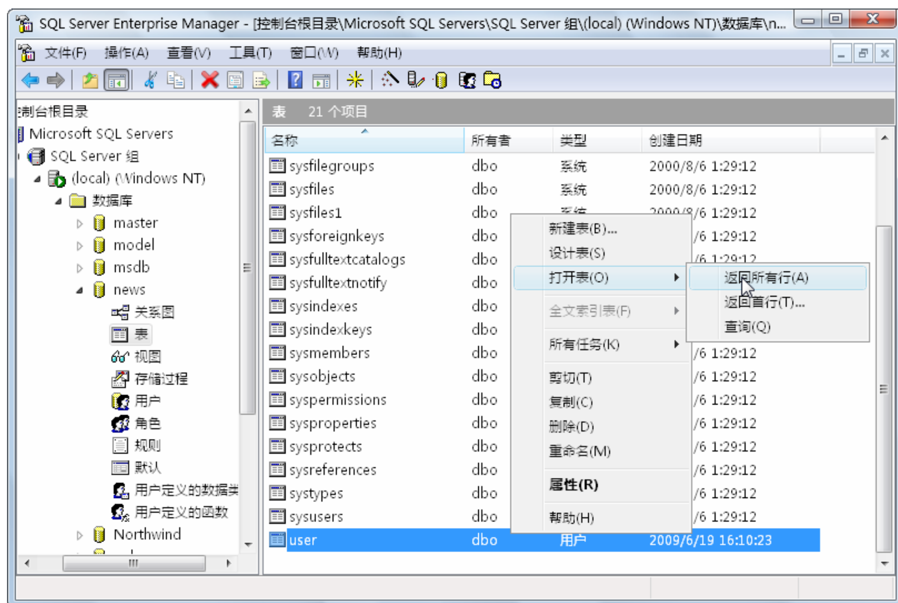


图 16-39　返回所有行命令

2）在打开的新窗口中为不同的字段添加值，如 ID 字段填写“|”，name 字段填写“maomo”，如图 16-40 所示。

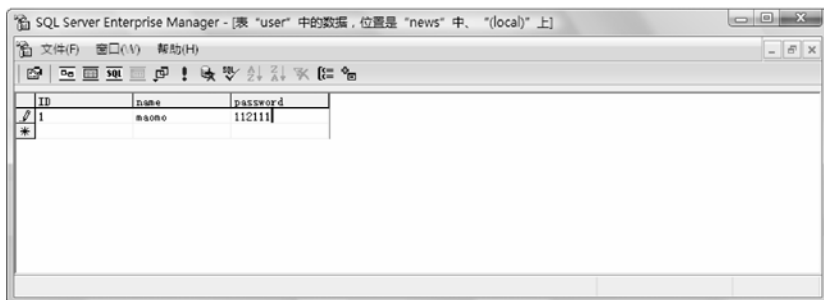


图 16-40　输入记录

16.6　疑难问题解析

本章详细介绍了数据库的定义、MySQL 和 SQL Server 的安装与使用。本节将对本章中较难理解的问题进行讲解。

读者疑问：除了 MySQL、SQL Server 数据库软件，还有其他数据库软件吗？哪一款数据库软件比较优秀呢？

解答：数据库软件很多，除了本章讲的 MySQL、SQL Server 数据库软件外，还有 DB2、Oracle、Access 都是有名的数据库软件，用户在选择数据库软件时，需考虑其特点，根据其需要进行选择。MySQL 是免费开源的，适用中、小型项目，SQL Server Oracle 是针对大型企业的超大型项目，价格较高，对计算机硬件配置也有一定的要求。

读者疑问：讲解 MySQL 时，讲解了 MySQL 的 SQL 语句。除了 MySQL，其他数据库的 SQL 语言是怎么一回事呢？是否相同呢？

解答：SQL 语句是数据库的一个标准，所有数据库的 SQL 语句大致相同。而 SQL 语句编程时可以分为 4 种，分别是数据查询语言（SELECT 语句）、数据操纵语言（INSERT, UPDATE, DELETE 语句）、数据定义语言（如 CREATE, DROP 等语句）和数据控制语言（如 COMMIT，ROLLBACK 等语句）。



职场点拨——我有一颗创业心

比尔盖茨和马云的神话离我们仿佛很近很近，当前一夜成名的典范有很多，不管是哪个行业，每年都会涌现出一批骄子和弄潮儿。但是作为一名程序员，如果你已经决定创业，并且义无反顾。那么一定要做好充分的准备，否则可能会功败垂成。下面是 7 条程序员创业的致命错误，希望能够对正在准备创业的程序员们起到一点警示作用。

1）程序员以年轻人居多，创业之初一般都有恢宏的目标，大多都喜欢大的排场。一心追求难度大的大生意，而忽视了数量多、难度小的小单子。但是到头来发现自己根本不具备

做大项目的能力，并且也超过了自己的经济承受能力。

2）初期创业投资者的共性是盲目追求大的投资规模，结果是造成了资产负债比率过高。如果一开始就把摊子铺得很大，忽视了种种危机的蛰伏，稍有不慎就可能爆发危机而影响全局。

3）情绪化的投资心态。都说失败乃成功之母，不去尝试就不会有收获，不去尝试，就不会积攒到经验。但是不害怕失败不等于支持鲁莽，初期创业者是无法承受屡屡创业失败的压力的，情绪化是最可怕的投资陷阱之一。如果你是一名即将创业的程序员，那么必须保证有清醒的头脑，冷静而客观地决策。

4）对合作缺乏真诚，频频违约。私心太重，合作缺乏诚意，不信守承诺，是投资合作中常见的事，亦常常成为投资失败的诱因。

5）思维受限制，不能立足长远，总想赚快钱，寻找短平快项目。

6）过度相信他人，不亲自进行市场调查。都说耳听为虚眼见为实，别人只是旁观者，他们看到的往往是冰山一角。你进行的创业关乎着你的命运，所以你要从多个角度进行调研，需要收集完整的资料后才能做决定。

7）只求盈利，不进行创新。

所以无论是即将要创业的读者，还是已经创业的读者，在此笔者建议你们一定要避免上述错误，祝你们早日踏上创业的成功之路。

第 17 章　数据库编程

数据库将所有的信息存储起来，就好比一个仓库，Java 可以从这个数据仓库中取出所需要的东西，也可以将外部的信息存储到数据库中去。Java 要操作数据库，必须要使用 SQL 语句，数据库编程实际就是使用 SQL 语言和 Java 语言操作数据库的程序设计，本章将详细讲解数据库编程的知识。本章主要内容如下：

- 什么是 JDBC。
- 连接数据库。
- SQL 语句。
- 职场点拨——团队结构看升职。

2012 年 XX 月 XX 日，多云

今天接到了在 Linux 系统上开发一个采购系统的任务，目前公司的每个项目都处在关键期，无法抽调其他项目经理负责这个项目。



一问一答

小菜：“公司有个 Linux 项目，需要项目经理独立完成，但目前所有项目经理都有任务在身，我很想试试！”

Wisdom：“恩，这是一个升职的机会！因为涉及了 Linux，只要你顺利完成任务，肯定会升职的！”

小菜：“真的吗？”

Wisdom：“试想作为同级别的同事，编程的技术水平本身都差不多，只有从深度和偏度上考虑才能脱颖而出。在深度上 Linux 是最佳选择，定会让你在领导面前发光出彩的。”

小菜：“明白了，那我就借你吉言了！言归正传，本章的数据库编程应该算是 Java 体系的核心应用之一吧？”

Wisdom：“是的，数据库改变了软件行业命运，无论是从技术角度还是应用范围来看，都在 Java 编程领域占据了绝对的制高点”

17.1　SQL 操作

在 MySQL 数据库里创建一个 student 的数据库，然后创建一个 student 的表，其表的结构如图 17−1 所示。

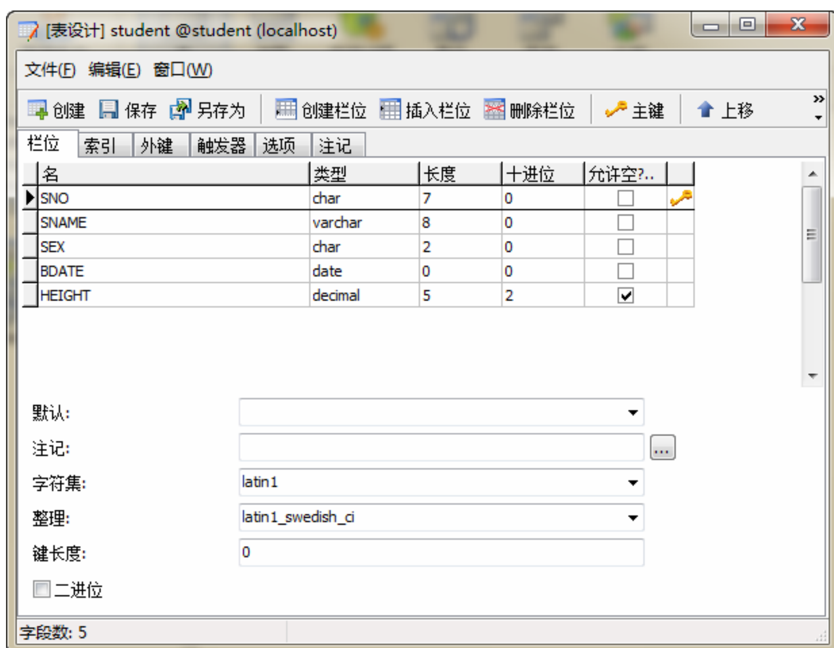


图 17-1　表设计

输入记录，然后单击选中表，单击鼠标右键，选择“转储 SQL 语句”命令，打开这个文件，得到下面的代码：

```
/*
MySQL Data Transfer
Source Host:localhost
Source Database:student
Target Host:localhost
Target Database:student
Date:2009/8/10 9:18:56
*/

SET FOREIGN_KEY_CHECKS=0;
-- ----------------------------
-- Table structure for student
-- ----------------------------
DROP TABLE IF EXISTS`student`;
CREATE TABLE`student` (
  `SNO` char(7) CHARACTER SET utf8 NOT NULL,
  `SNAME` varchar(8) NOT NULL,
  `SEX` char(2) NOT NULL,
  `BDATE` date NOT NULL,
  `HEIGHT` decimal(5,2) DEFAULT '0.00',
  PRIMARY KEY (`SNO`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
-- ----------------------------
-- Records
-- ----------------------------
INSERT INTO `student` VALUES ('1','titi','8','2000-06-23','145.00');
INSERT INTO `student` VALUES ('2','gougou','7','2001-05-27','150.00');
```

上面的代码就是 SQL 语句，利用它可以操作数据库，在这里用户只需要明白什么是 SQL 语句就可以了。

17.2 什么是 JDBC

JDBC 就是 Java 连接数据库的一个工具，没有这个工具，Java 将没有办法连接数据库，用户只需对它有一定了解即可，不用具体学习 JDBC 的结构。

17.2.1 JDBC API

JDBC 是个“低级”接口，它用于直接调用 SQL 命令。在这方面它的功能非常出色，使数据库连接 API 更易于操作，但它同时也被设计为一种基础接口，在它之上还可以建立更高级的接口和工具。高级接口是“对用户友好的”接口，它使用的是一种更易理解和更为方便的 API，这种 API 在幕后被转换为诸如 JDBC 这样的低级接口。

在关系数据库的“对象/关系”映射中，表中的每行对应于类中的一个实例，而每列的值对应于该实例的一个属性。于是，程序员可直接对 Java 对象进行操作。存取数据所需的 SQL 调用将在“掩盖下”自动生成。此外还可提供更复杂的映射，例如将多个表中的行合并到一个 Java 类中。

随着人们对 JDBC 的兴趣地提高，越来越多的开发人员开始使用基于 JDBC 的工具，以使程序的编写更加容易。程序员也一直在编写力图使最终用户对数据库的访问变得更为简单的应用程序。例如应用程序可提供一个选择数据库任务的菜单。任务被选定后，应用程序将给出提示及空白填写执行选定任务所需的信息。所需信息输入应用程序将自动调用所需的 SQL 命令。在这样一种程序的协助下，即使用户根本不懂 SQL 的语法，也可以执行数据库任务。

17.2.2 JDBC 驱动类型

JDBC 是应用程序编程接口，描述了一套访问关系数据库的标准 Java 类库，并且还为数据库厂商提供了一个标准的体系结构，让厂商可以为自己的数据库产品提供 JDBC 驱动程序，这些驱动程序可使 Java 应用程序直接访问厂商的数据产品，从而提高了 Java 程序访问数据库的效率。在 Java 程序设计中，JDBC 驱动可以分为以下四种：

- JDBC-ODBC 桥：ODBC 是微软公司开放服务结构(WOSA，Windows Open Services Architecture)中有关数据库的一个组成部分，它建立了一组规范，并提供了一组对数据库访问的标准 API（应用程序编程接口）。这些 API 利用 SQL 来完成其大部分任务。ODBC 本身也提供了对 SQL 语言的支持，用户可以直接将 SQL 语句送给 ODBC，因为

ODBC 推出的时间要比 SQL 早，所以大部分数据库都支持通过 ODBC 来访问。SUN 公司提供了 JDBC-ODBC 这个驱动来支持像 Microsoft Access 这样的数据库，JDBC API 通过调用 JDBC-ODBC 桥，而 JDBC-ODBC 又调用了 ODBC API，从而达到访问数据库的 ODBC 层，这种方式经过了多层调用导致效率比较低，用这种方式访问数据库，需要用户的计算机中装有 JDBC-ODBC 驱动、ODBC 驱动和相应的数据库的本地 API。

- 本地 API 驱动：本地 API 驱动直接把 JDBC 调用转变为数据库的标准调用再去访问数据库，这种方法需要本地数据库驱动代码。本地 API 驱动—厂商 DB 代码数据库 Server ，这种驱动比起 JDBC-ODBC 执行效率大大提高了。但是，它仍然需要在客户端加载数据库厂商提供的代码库，这样就不适合基于 internet 的应用。并且，他的执行效率比起 3、4 型的 JDBC 驱动还是不够高。
- 网络协议驱动：这种驱动实际上是根据我们熟悉的三层结构建立的。JDBC 先把对数据库的访问请求传递给网络上的中间件服务器。中间件服务器再把请求翻译为符合数据库规范的调用，再把这种调用传给数据库服务器。如果中间件服务器也是用 Java 开发的，那么在中间层也可以使用 1、2 型 JDBC 驱动程序作为访问数据库的方法。网络协议驱动——中间件服务器数据库 Server，由于这种驱动是基于 server 的，所以它不需要在客户端加载数据库厂商提供的代码库。而且他在执行效率和可升级性方面也是比较好的。因为大部分功能实现都在 server 端，所以这种驱动可以设计的很小，可以非常快速的加载到内存中。但是，这种驱动在中间件中仍然需要有配置数据库的驱动程序，并且由于多了一个中间层传递数据，它的执行效率还不是最好。
- 本地协议驱动：这种驱动直接把 JDBC 调用转换为符合相关数据库系统规范的请求。由于 4 型驱动编写的应用可以直接和数据库服务器通讯，这种类型的驱动完全由 Java 实现，本地协议驱动数据库 Server，由于这种驱动不需要先把 JDBC 的调用传给 ODBC 或本地数据库接口或者是中间层服务器，所以它的执行效率是非常高的。而且，它根本不需要在客户端或服务器端装载任何的软件或驱动。这种驱动程序可以动态的被下载，但是对于不同的数据库需要下载的驱动程序也不同。

17.3　连接数据库

要操作数据库，首先就必须要连接数据库，本节将详细讲解如何连接 MySQL 和 SQL Sever 数据库，对于其他数据库用户也可以举一反三。

17.3.1　轻松连接 MySQL

连接 MySQL 可以通过下面的几个步骤完成，要连接 MySQL，首先需要下载 JDBC 驱动，然后再进行连接。

（1）下载 MySQL 驱动

用户可以通过搜索引擎下载 MySQL 驱动，如果用户对英文比较熟悉，不妨碍去官方网站下载，如图 17−2 所示。



图 17-2 下载 JDBC 驱动

（2）配置 MySQL 驱动

下载驱动后，将其解压，找到 mysql-connector-java-5.1.7-bin.jar 的文件，将其放置到一个文件中，如果是使用 dos 命令执行 Java 程序，就必须要对环境进行配置，其具体操作如下：

1）在桌面上单击选中计算机图标，单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，然后单击左侧的“高级系统设置”连接，在弹出的窗口中单击打开“高级”选项卡，然后单击 环境变量(N)... 按钮，如图 17-3 所示。

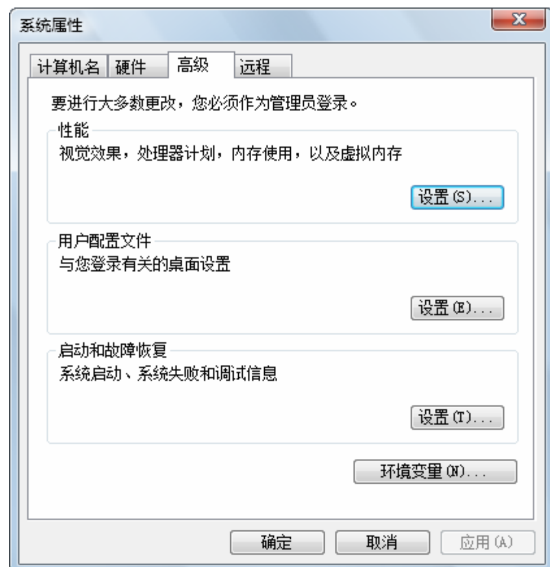


图 17-3 “系统属性”窗口

2）在前面的章节中曾经讲解过如何配置 JDK，并建立了一个 CLASSPATH 环境，现在要找到这个环境变量，单击 编辑(I) 按钮重新对它进行编辑，如图 17-4 所示。

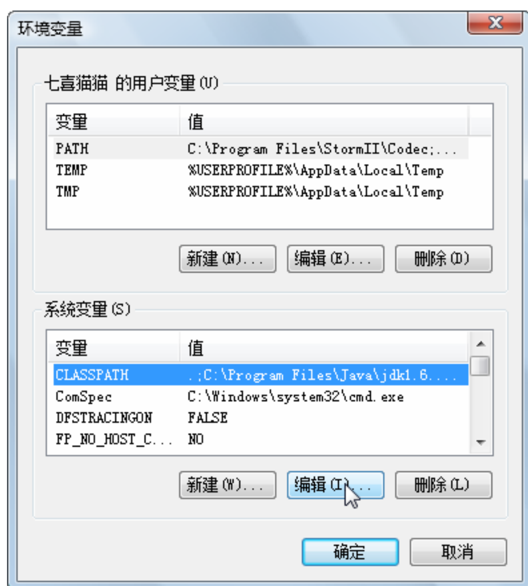


图 17-4　编辑 classpath

3）在它的变量值后面添加“;”，然后再添加 mysql-connector-java.jar 文件的路径，该文件放置 D 盘，所以为“; D:\mysql-connector-java-5.1.7-bin.jar”，单击确定按钮，然后再次单击确定按钮，如图 17-5 所示。

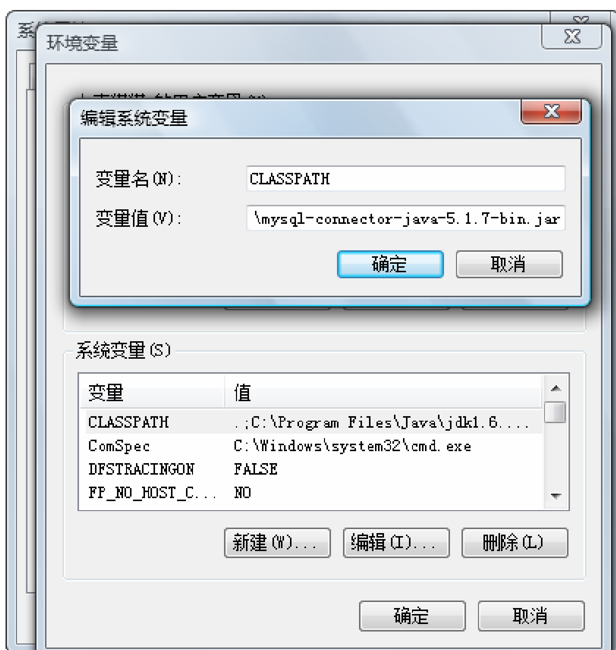


图 17-5　编辑系统变量

提示：上面讲解的只是 Java 连接 MySQL 的一个原理，在实际中并不适用，因为现在都使用 IDE 开发 Java 项目，用户将 JDBC 驱动直接加载到 IDE 里即可。

（3）将 MySQL 驱动加载到 Eclipse

在实际开发的过程中，很少有人会使用这种方式开发，而是用 Eclipse 或 MyEclipse 开发 Java 程序，Eclipse 和 MyEclipse 的驱动配置是一样的，下面以 Eclipse 为例进行配置，其具体操作如下：

1）启动 Eclipse，选中下载的驱动文件，单击鼠标右键，选择“复制”命令，然后选择需要的项目，如图 17-6 所示。

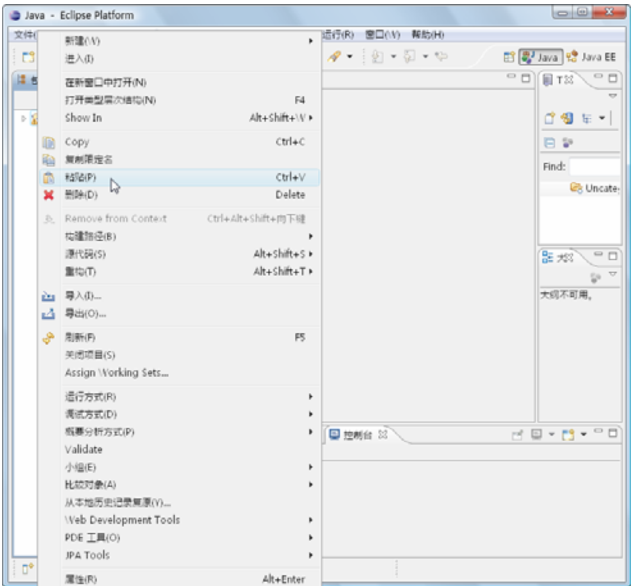


图 17-6　复制与粘贴

2）选中加载的驱动，单击鼠标右键，在弹出的快捷菜单中选择“构建路径/添加至构建路径”命令，如图 17-7 所示。

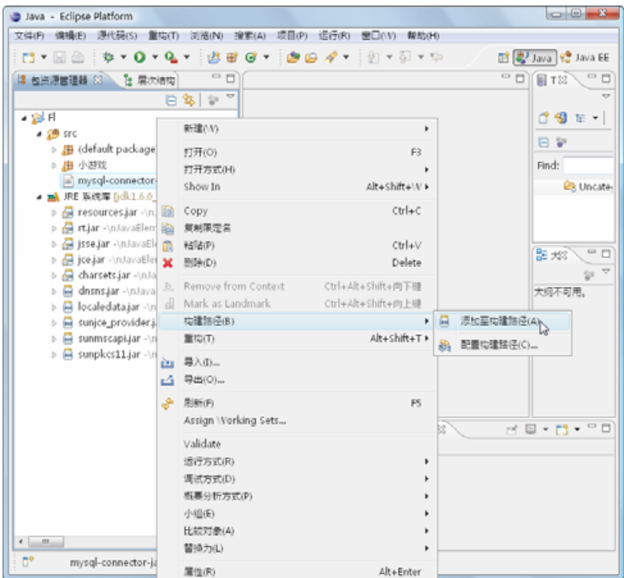


图 17-7　选择命令

（4）测试连接

配置完成后，还需要一个程序去连接一个数据库。下面将编写一个程序去连接开篇所展示的数据库，其代码见“光盘：源代码/第 17 章/Lian.java”：

```
import java.sql.*;
public class Lian
{
public static void main(String[] args)
{

  // 驱动程序名
           String driver="com.mysql.jdbc.Driver";
           // URL 指向要访问的数据库名 scutcs
           String url="jdbc:mysql://127.0.0.1:3306/student";
           // MySQL 配置时的用户名
           String user="root";
             // MySQL 配置时的密码
           String password="1234";
           try {
            // 加载驱动程序

            Class.forName(driver);
 // 连续数据库
Connection conn=DriverManager.getConnection(url, user, password);
            if(!conn.isClosed())
System.out.println("Succeeded connecting to the Database!");
            // statement 用来执行 SQL 语句
            Statement statement=conn.createStatement();
            // 要执行的 SQL 语句
            String sql="select * from student";
            // 结果集
            ResultSet rs=statement.executeQuery(sql);
            System.out.println("-----------------");
            System.out.println("执行结果如下所示:");
            System.out.println("-----------------");
            System.out.println(" 学号" + "\t" + " 姓名"+"\t"+"出生日期");
            System.out.println("-----------------");
            String name=null;
            while(rs.next())
{

                // 选择 sname 这列数据
             name=rs.getString("sname");
    // 首先使用 ISO-8859-1 字符集 name 解码为字节序列并将结果存储到新的字节数组中。
             // 然后使用 GB2312 字符集解码指定的字节数组
             name=new String(name.getBytes("ISO-8859-1"),"GB2312");
```

```
                    // 输出结果
                    System.out.println(rs.getString("sno") + "\t" + name);
                  }
                  rs.close();
                  conn.close();
                }
        catch(ClassNotFoundException e)
        {
                  System.out.println("Sorry,can`t find the Driver!");
                  e.printStackTrace();
                } catch(SQLException e)
         {
                  e.printStackTrace();
                }
        catch(Exception e)
        {
                  e.printStackTrace();
                }
        }
        }
```

运行代码，得到如图 17-8 所示的结果。

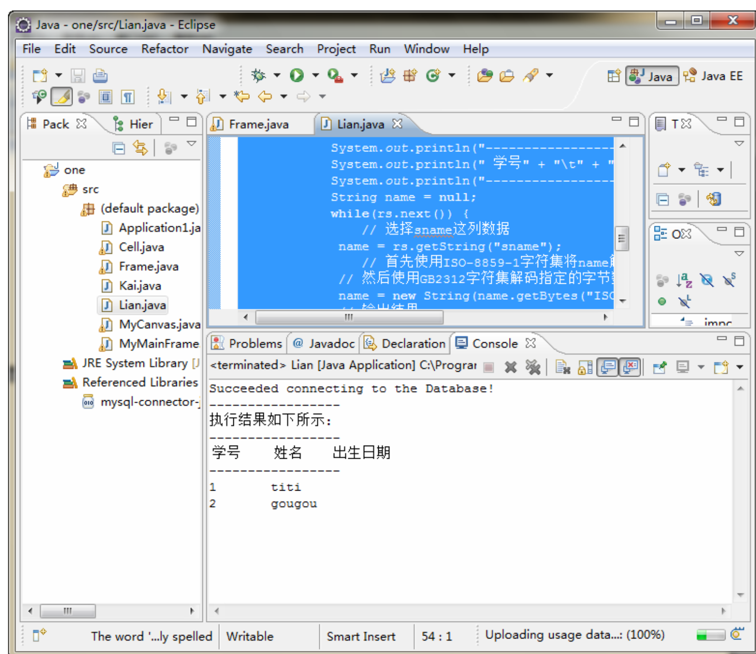


图 17-8 连接 MySQL 成功

17.3.2 轻松连接 SQL Sever 2000

连接 SQL Sever 2000 和连接 MySQL 十分相似，但是也有不同之处，本节将进行详细讲解。

（1）安装 SQL Sever 2000 驱动

用户可以通过搜索引起搜索 SQL Server 2000 Drive for JDBC SP3 驱动程序，下载并解压后即可进行安装。安装过程十分简单，其具体操作如下：

1）双击下载的 SQL Server 2000 Driver for JDBC SP3.exe 文件，打开如图 17−9 所示的窗口。

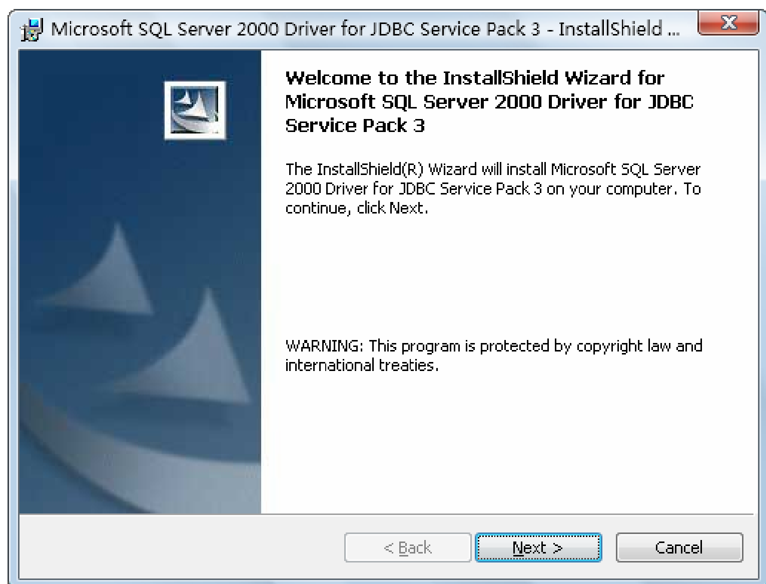


图 17−9　第一个向导

2）单击“Next”按钮，在弹出的协议对话框中选择第一个复选框，即同意协议内容，然后单击“Next”按钮，如图 17−10 所示。



图 17−10　协议对话框

3）在打开的窗口中用户按照默认设置即可，然后单击“Next”按钮，如图 17−11 所示。

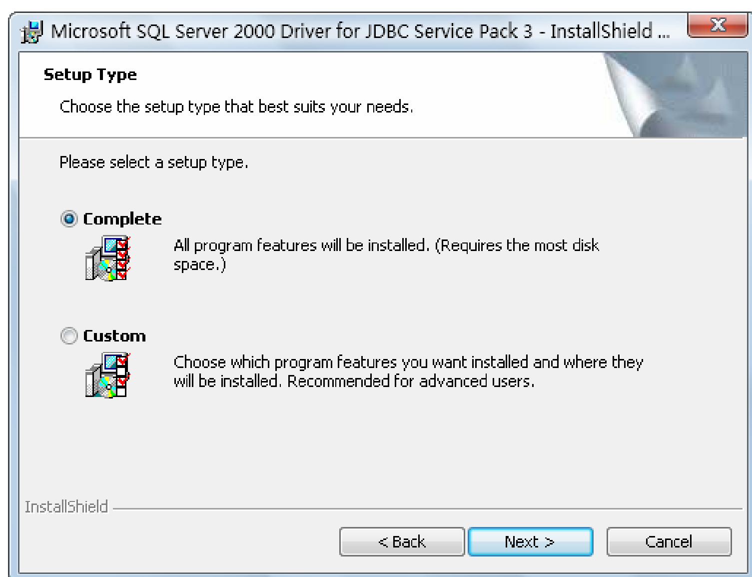


图 17−11　按照默认设置

4）在打开的窗口中单击“Install”按钮，即可进行安装。完成后将会打开一个完成安装的窗口，单击“Finish”按钮即可完成安装，如图 17−12 所示。

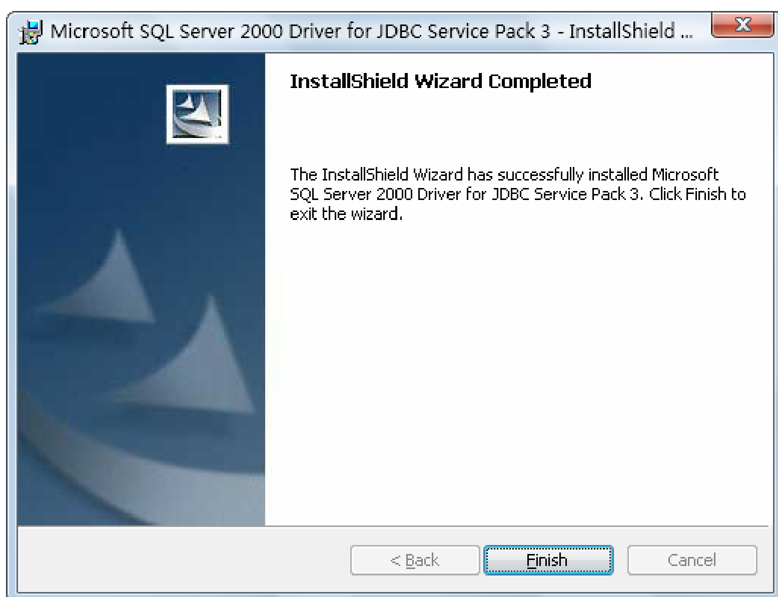


图 17−12　完成安装

（2）将 SQL Sever 2000 驱动加载到 Eclipse 里

在安装 SQL Sever 2000 的根目录中的 lib 文件夹下，将三个 jar 文件进行复制，然后对它进行配置，其具体操作操作如下：

1）启动 Eclipse，选中下载的驱动文件，单击鼠标右键，选择“粘贴”命令，然后选择需要的项目，如图 17-13 所示。

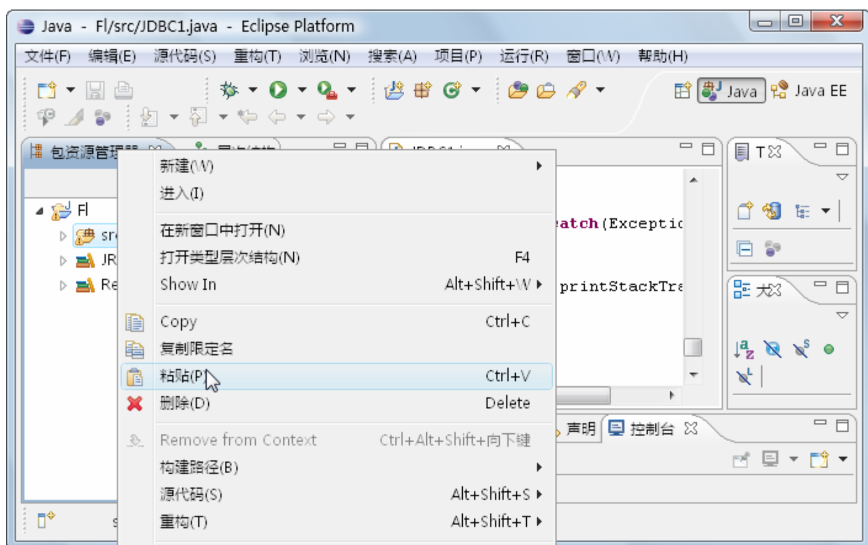


图 17-13　粘贴 JAR 文件

2）选中加载的驱动，单击鼠标右键，在弹出的快捷菜单中选择“构建路径/添加至构建路径”命令，如图 17-14 所示。

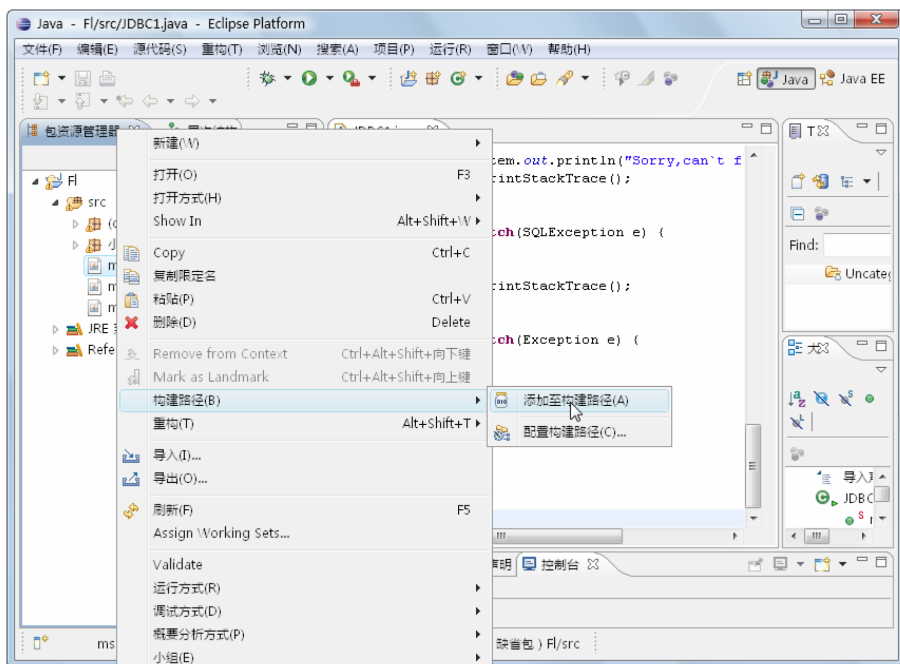


图 17-14　添加到构建路径

3）用相同的方法添加三个 jar 文件，最后得到如图 17-15 所示的结果。

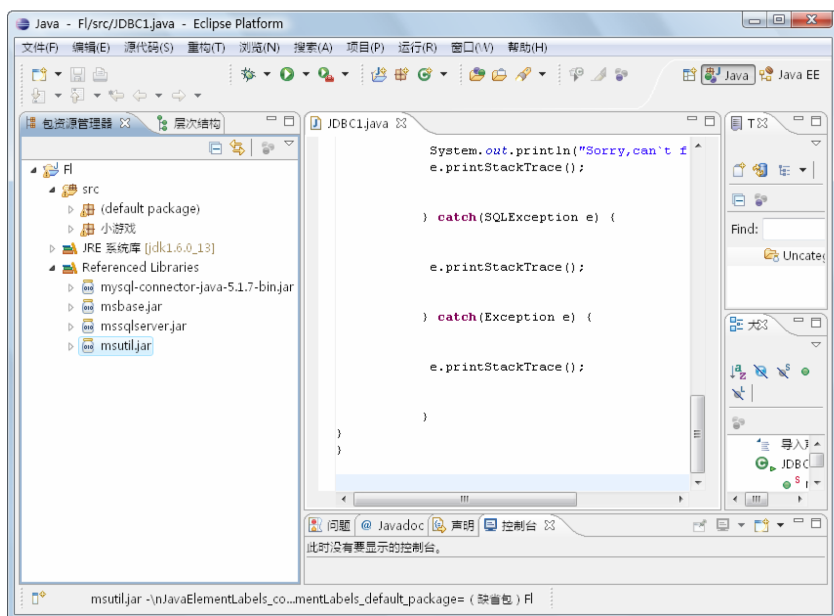


图 17-15　添加三个 jar 文件

（3）测试连接

加载好驱动后就可以通过编写一个程序进行测试。下面就编写一个用来连接数据库服务器上的 pubs 的程序，其代码见“光盘：源代码/第 17 章/TestDB.java”：

```
import java.sql.*;
public class TestDB
{
public static void main(String[] args)
{

String driverName="com.microsoft.jdbc.sqlserver.SQLServerDriver"; //加载驱动
String  dbURL="jdbc:microsoft:sqlserver://localhost:1433;  DatabaseName=pubs";
//连接数据库
String userName="sa"; //服务器用户名
String userPwd=""; //用户名密码
Connection dbConn;
try {
Class.forName(driverName);
dbConn=DriverManager.getConnection(dbURL, userName, userPwd);
System.out.println("连接成功");
}
catch (Exception e) {
e.printStackTrace();
}
```

```
        }
    }
```

运行代码，得到如图 17-16 所示的结果。

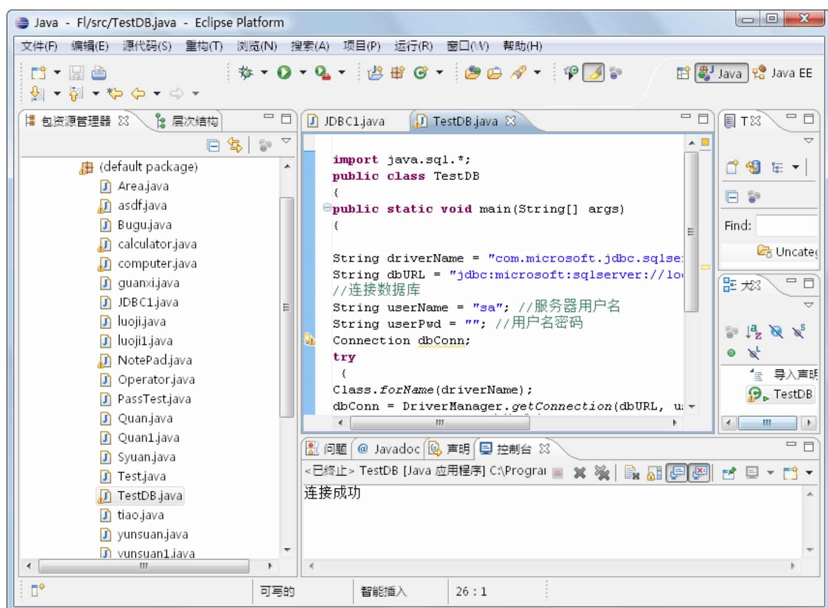


图 17-16　连接成功

17.4　SQL 语句

编写程序绝对离不开 SQL 语句，下面将讲解常用的 SQL 语句。

17.4.1　新建数据库表

在使用 SQL 语句操作数据库的过程中，数据库定义的语言主要是新建、删除等语句，创建数据库、创建表的 SQL 语句的格式如下：

```
    CREATE DATABASE `china` ;
```

参数介绍如下：

- ❑ CREATE：关键字。
- ❑ DATABASE：数据库新建的关键字
- ❑ china：数据库名。

表格是数据库中存储资料的基本架构，是用来表示数据的完美形式。在表格中 1 行通常称做记录，列称做字段，通过程序建立表通常是建立表的名称以及包的内容，即字段、字段属性。其语法格式如下：

```
    CREATE TABLE "表格名"
    ("栏位 1" "栏位 1 资料种类",
```

```
"栏位 2" "栏位 2 资料种类"
... )
```

下面新建一个表格，其代码如下：

```
CREATE TABLE customer
(First_Name char(50),
Last_Name char(50),
Address char(50),
City char(50),
Country char(25),
Birth_Date date)
```

运行代码，得到如图 17-17 所示的结果。



图 17-17 创建的表

在 SQL 语句中，还提供了一个 DROP TABLE 的语法来让我们清除表格，该语法的格式如下：

```
DROP TABLE table_name
```

其参数介绍如下：

- DROP：关键字。
- TABLE：数据库新建的关键字
- table_name：数据库名。

17.4.2 数据库查询语句

在操作数据库时，会频繁地使用数据库查询语句，下面将详细讲解如何查询和查询的一些技巧。

（1）SELECT 语句

SELECT 语句是用来做什么的呢？它是用来在数据库表中检索数据行和列的。表格是一个数据库内的基本结构，它的目的是存储资料。在表格处理这一部分中，我们会提到如何使用 SQL 来设定表格，设定的格式如下：

```
SELECT "栏位名" FROM "表格名"
```

下面列举一个表，其表（Store_Information 表格）的内容如表 17-1 所示。

表　17-1

store_name	Sales	Date
Los Angeles	$1500	Jan-05-1999
San Diego	$250	Jan-07-1999
Los Angeles	$300	Jan-08-1999
Boston	$700	Jan-08-1999

倘若要选出所有的店名 (store_Name)，只需输入下面的语句：

```
SELECT store_name FROM Store_Information
```

执行这个语句，就会得到下面的结果。

store_name

Los Angeles

San Diego

Los Angeles

Boston

（2）使用 WHERE 语句查询

使用 WHERE 语句设置查询条件，可以过滤掉不需要的数据。例如下面这段代码，就是使用 WHERE 语句查询年龄大于 50 的数据：

```
SELECT *
FROM usertable
WHERE age>50
```

WHERE 语句可包括各种条件运算符，其介绍如下：

- 比较运算符（大小比较）：>、>=、=、<、<=、<>、!>、!<。
- 范围运算符（表达式值是否在指定的范围）：BETWEEN…AND…。
- NOT BETWEEN…AND…。
- 列表运算符（判断表达式是否为列表中的指定项）：ＩＮ（项 1,项 2……）。
- NOT IN（项 1,项 2……）。
- 模式匹配符（判断值是否与指定的字符通配格式相符）：LIKE、NOT LIKE。
- 空值判断符（判断表达式是否为空）：IS NULL、NOT IS NULL。
- 逻辑运算符（用于多条件的逻辑连接）：NOT、AND、OR。

在逻辑逻辑运算符中，又分为以下几种情况：

- 范围运算符例：age BETWEEN 10 AND 30 相当于 age>=10 AND age<=30。
- 列表运算符例：country IN ('Germany','China')。
- 模式匹配符例：常用于模糊查找，它判断值是否与指定的字符串格式相匹配。可用于 char、varchar、text、ntext、datetime 和 smalldatetime 等类型查询。

还可使用以下通配字符：

百分号“%”：可匹配任意类型和长度的字符，如果是中文，需使用两个百分号%%。

下画线“_”：匹配单个任意字符，它常用来限制表达式的字符长度。

方括号“[]”：指定一个字符、字符串或范围，要求所匹配对象为它们中的任一个。

[^]：其取值也和[] 相同，但它要求所匹配对象为指定字符以外的任一个字符。

限制以 Publishing 结尾：使用 LIKE '%Publishing'。

限制以 A 开头：LIKE '[A]%'。

限制以 A 开头外：LIKE '[^A]%'。

- 空值判断符 WHERE age IS NULL。
- 逻辑运算符：优先级为 NOT、AND、OR。

17.4.3 数据库操纵语句

本节将讲解如何操作 SQL 数据库，主要讲解插入记录、修改和删除等操作。

（1）INSERT 语句添加行

该语句主要用于插入记录，换句话讲，它就是用于向表中添加行。在 INSERT 语句中，可以指定以下信息：

- 要插入的行所在的表。
- 要为其指定值的列的列表。
- 要为这些列指定的值的列表。

在添加行时，通常需要为主键和其他被定义为 NOT NULL 的列指定值。如果不想，也可以不为定义为 NULL 的列指定值。默认情况下，这些列都会被设置为空值。

在为一个表添加记录时，主要分为省略列的列表、为列指定空值、在值中使用单引号和双引号、从一个表复制到另一个表的记录这几个步骤。

当为所有的列都提供值时可以省略列的列表，例如：

```
INSERT INTO customers
VALUES (7,'Jane','Green','01-JAN-1970','800-555-1216');
```

当省略列的列表时，指定的值顺序必须与 DESCRIBE 命令输出结果中显示的列的顺序一致。

（2）使用 UPDATE 语句修改行

UPDATE 语句用于修改表中的数据，主要用于修改行，其格式如下：

```
UPDATE 表名称 SET 列名称=新值 WHERE 列名称=某值
```

假设有这样一个表 Person，如表 17-2 所示。

表 17-2

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing
Wilson		Champs-Elysees	

现在要更换其中一行的数据，为 LastName 是“Wilson”的人添加 FirstName，可以使用下面的 SQL 语句：

```
UPDATE Person SET FirstName='Fred' WHERE LastName='Wilson'
```

修改后结果如表 17-3 所示。

表　17-3

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing
Wilson	Fred	Champs-Elysees	

上面这个例子只是更改了其中的一个值，若需要同时更改若干个值，则需使用下面的 SQL 语句修改地址（address），并同时添加城市名称（city），语句如下：

```
UPDATE Person SET Address='Zhongshan 23', City='Nanjing'
WHERE LastName='Wilson'
```

修改后结果如表 17-4 所示。

表　17-4

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing
Wilson	Fred	Zhongshan 23	Nanjing

（3）删除语句 DELETE

DELETE 主要用于删除行，其格式如下：

```
DELETE FROM 表名称 WHERE 列名称 = 值
```

下面展示一张表，如表 17-5 所示。

表　17-5

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing
Wilson	Fred	Zhongshan 23	Nanjing

倘若要删除其中的一行，使用下面的语句即可：

```
"Fred Wilson" 会被删除：
DELETE FROM Person WHERE LastName='Wilson'
```

运行上面这段 SQL 语句后，得到如表 17-6 所示的结果。

表　17-6

LastName	FirstName	Address	City
Gates	Bill	Xuanwumen 10	Beijing

使用下面的两种语句可以在不删除表的情况下删除所有的行，这意味着表的结构、属性和索引都是完整的：

```
DELETE FROM table_name
```

或者：

```
DELETE * FROM table_name
```

（4）使用索引 INDEX

索引（Index）可以帮助用户从表格中快速找到需要的资料。举例来说，假设要在一本园艺书中找到关于种植青椒的信息，若这本书没有索引的话，就只有从头开始读，直到找到有关种青椒的地方为止。若这本书有索引的话，就可以先查看索引，找出种植青椒的资讯的页码，然后直接翻到该页即可。很明显，运用索引是一种有效且省时的方法。因此，在表格上建立索引是一件有利于提高系统效率的事。一个索引可以涵盖一个或多个行位，建立索引的语法格式如下：

```
CREATE INDEX "INDEX_NAME" ON "TABLE_NAME" (COLUMN_NAME)
```

假设有这样一个表格，代码如下：

```
TABLE Customer
(First_Name char(50),
Last_Name char(50),
Address char(50),
City char(50),
Country char(25),
Birth_Date date)
```

若要在 Last_Name 列上建立一个索引，可以使用如下代码：

```
CREATE INDEX IDX_CUSTOMER_LAST_NAME
on CUSTOMER (Last_Name)
```

若要在 Last_Name 这个栏位上建立一个索引，可以使用如下代码：

```
CREATE INDEX IDX_CUSTOMER_LOCATION
on CUSTOMER (City, Country)
```

索引的命名并没有一个固定的方式。最常用的方式是在名称前加一个字首，例如"IDX_"，以此来避免与资料库中的其他事物混淆。另外，在索引名之内包括表格名及行位名也是一个好的方式。

提示：每个资料库会有它本身的 CREATE INDEX 语法，而不同资料库的语法也会有所不同。因此在下指令前，应先在资料库使用手册中确认正确的语法。

17.5 疑难问题解析

本章详细介绍了 SQL，连接数据库、SQL 语句等基本知识。本节将对本章中较难理解的问题进行讲解。

读者疑问：在 Java 编写过程中，用 Java 窗口程序连接数据库与 Java Web 连接数据库相同吗？

解答：大致相同，在 Java Web 编程过程中，大部分功能都是通过调用 Class 文件来实现的，用户可以使用 JavaBean 的方法连接数据库。

读者疑问：在数据库编程中，是如何实现让 Java 程序操作数据库这一功能的呢？

解答：在权限允许的情况下，SQL 几乎可以操作数据库的全部功能，Java 数据库编程实际上是 SQL 语句的功劳，Java 程序只负责数据的上层处理，不能直接操作数据库。



职场点拨——谈加薪升职

每一位在公司里工作的职场人士都期望赢得加薪升职的机会，程序员也不例外。当前竞争日益激烈，怎样才能在众多竞争者中脱颖而出，成功升职呢？笔者在此有如下三条建议供大家借鉴。

（1）脱颖而出，让老板惊喜

惊喜有很多种，例如提前完成了项目，例如给公司介绍了一个新客户。杰克·韦尔奇曾透露这样一条升职秘诀："我想要做的就是'脱颖而出'。如果我仅仅回答了老板的问题，那么就很难引起注意。其实每当老板们提出问题时，他们在脑海中早已经有了自己的答案，他们只是想得到再次确认而已。为了显示与众不同，我想我的回答应该比提出的问题范围更广一些。我想给出的不仅仅是答案，还有意料之外的新鲜的观点。"就是遵循这条原则，杰克·韦尔奇从通用电气最底层开始打拼并最终成为通用电气首席执行官。

（2）懂得办公室游戏规则

当前竞争日益激烈，究竟怎样才能在现代职场中立足并脱颖而出呢？美国职业咨询专家给出的答案是："经理们依旧倾向于雇用和提拔与之相处和谐的人，也就是那些聪明的，懂得办公室政治的人。"无论什么游戏，都有它自己的规则，程序员所处的办公室也不例外。程序员也要遵循游戏规则，如果不遵循，升职就变得遥遥无期，薪水就会原地踏步，老板就会对你熟视无睹，甚至被老板炒鱿鱼。当然实力才是最坚固的发展基础，如果您没有专业实力，那么即使拥有再高超的办公室政治技巧，也终将一败涂地。

（3）掌握说服老板的最核心法则

富兰克林曾经说过："如果你想说服某人，不要诉诸于道德，而要诉诸于利益。"我们可以将这条规则运用在加薪升职领域。例如程序员 AA 这样说服老板给他加薪，"从整个大环境来评估，去年我每个月的平均所得有些差强人意。但我觉得自己绝对物超所值，应该有更高的薪金，我已经列出去年所经手的项目和客户满意度，以此来向公司证明自己的工作表现。"后来 AA 决定向老板列举为公司所赚取的各种利润，以及旗下客户愿意继续合作的稳定度分析，并最终决定向老板提出加薪 10％的要求，结果很出乎 AA 的意料，因为公司竟然足足给他加了 15％的薪水。

温故而知新——第三篇实战范例

第三篇学习的重点是窗口和数据库，本节将对本篇的知识进行全面回顾，使读者对这些知识的理解得到升华。

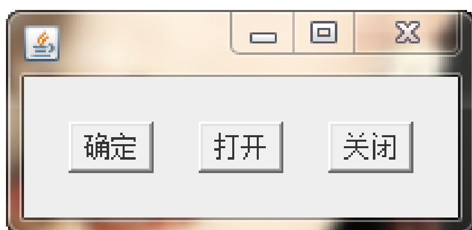
范例 1　顺序布局

顺序布局（Flow Layout）是最基本的一种布局方式，窗口的默认布局就是顺序布局。顺序布局指的是把图形元件一个接一个地放在窗口上。下面通过一段代码进行回顾，其代码见“光盘：源代码/温故而知新 3/ MyFlowLayout.java”：

```
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
public class MyFlowLayout
{
    private Frame f;
    private Button button1, button2, button3;
    public static void main (String args[])
 {
        MyFlowLayout mflow=new MyFlowLayout ();
        mflow.go();
    }
    public void go(){
        f=new Frame ("布局");
        f.addWindowListener(new WindowAdapter()
{
            public void windowClosing(WindowEvent evt)
 {
                f.setVisible(false);
                f.dispose();
                System.exit(0);
            }
        });
        //f.setLayout(new FlowLayout());
        f.setLayout(new FlowLayout(FlowLayout.LEADING, 20, 20));
        button1=new Button("确定");
        button2=new Button("打开");
        button3=new Button("关闭");
```

```
                f.add(button1);
                f.add(button2);
                f.add(button3);
                f.setSize (200,200);
                f.pack();
                f.setVisible(true);
          }
       }
```

运行代码，得到如范例图 3-1 所示的结果。



范例图 3-1　顺序布局

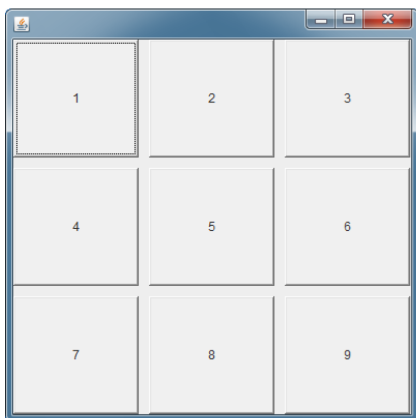
范例 2　网格布局

网格布局（GridLayout）把窗口分成一个个大小相等的网格，可以在网格中放置组件列数。下面通过一段代码进行回顾，其代码见“光盘：源代码/温故而知新 3/ MyGridLayout.java”：

```
       import java.awt.*;
       import java.awt.event.*;
       public class MyGridLayout
        {
           private Frame f;
           private Button[] btn;
           public static void main(String args[])
       {
                MyGridLayout grid=new MyGridLayout();
                grid.go();
           }
           public void go()
        {
                f=new Frame("GridLayout");
            f.addWindowListener(new WindowAdapter(){
                    public void windowClosing(WindowEvent evt) {
                        f.setVisible(false);
                        f.dispose();
                        System.exit(0);
                    }
                });
```

```
        f.setLayout (new GridLayout (3,3,10,10));
            btn = new Button[9];
            for(int i=0;i<=8;i++) {
               int j=i+1;
               btn[i]=new Button("" + j);
               f.add(btn[i]);
            }
        // f.pack();
           f.setSize(400, 400);
           f.setVisible(true);
         }
     }
```

运行代码，得到如范例图 3-2 所示的结果。



范例图 3-2　网格布局

范例 3　Swing 窗口（一）

Swing 窗口是 AWT 的升级版，有了它可以满足设计者更多的要求。下面通过一段代码进行回顾，其代码见“光盘：源代码/温故而知新 3/Login.java”：

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
import javax.swing.WindowConstants;
public class Login extends JFrame
{
   private JLabel userLabel;
   private JLabel passLabel;
   private JButton exit;
   private JButton login;
```

```
    private JTextField userName;
    private JPasswordField userPassword;
    public Login()
    {
         setTitle("登录窗口");
         setBounds(300,200,400,350);
         getContentPane().setLayout(null);
         userLabel = new JLabel();
         userLabel.setText("用户名：");
         userLabel.setBounds(10,10,200,18);
         getContentPane().add(userLabel);
         userName=new JTextField();
         userName.setBounds(60,10,200,18);
         getContentPane().add(userName);
         passLabel = new JLabel();
         passLabel.setText("密   码：");
         passLabel.setBounds(10,50,200,18);
         getContentPane().add(passLabel);
         userPassword = new JPasswordField();
         userPassword.setBounds(60,50,200,18);
         getContentPane().add(userPassword);
         login = new JButton();
         login.setText("进入");
         login.setBounds(90,80,60,18);
         getContentPane().add(login);
         exit = new JButton();
         exit.setText("退出");
         exit.setBounds(170,80,60,18);
         getContentPane().add(exit);
         setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }
    public static void main(String[] args)
    {
         Login login=new Login();
         login.setVisible(true);
    }
  }
```

运行代码，得到如范例图 3-3 所示的结果。



范例图 3-3　登录窗口

范例 4　Swing 窗口（二）

下面再通过一段代码对 Swing 窗口进行深入分析，以加深读者对 Swing 这一重要工具的理解，其代码见“光盘：源代码/温故而知新 3/A.java”：

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class A extends JFrame implements ActionListener
{
 JPanel jp=new JPanel();
 JTextField text=new JTextField(10);
 JButton btn=new JButton("按钮");
 JRadioButton rb=new JRadioButton("单选按钮");
 ButtonGroup bg=new ButtonGroup();
 JCheckBox box=new JCheckBox("复选框");
 public A()
 {
  bg.add(rb);
  jp.setLayout(new FlowLayout());
  jp.add(text);
  jp.add(btn);
  jp.add(rb);
  jp.add(box);
  btn.addActionListener(this);
  rb.addActionListener(this);
  box.addActionListener(this);

  this.add(jp);
  this.setBounds(100,100,200,150);
  this.setVisible(true);
  this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  this.pack();
 }
 public void actionPerformed(ActionEvent e)
 {
  String s=e.getActionCommand();
  if(s.equals("按钮"))
  {
   text.setText(s);
  }
  else if(s.equals("单选按钮"))
  {
   text.setText(s);
  }
  else if(s.equals("复选框"))
```

```
            {
             text.setText(s);
            }
           }
           public static void main(String[] args)
           {
            new A();
           }
          }
```

运行代码，得到如范例图 3-4 所示的结果。

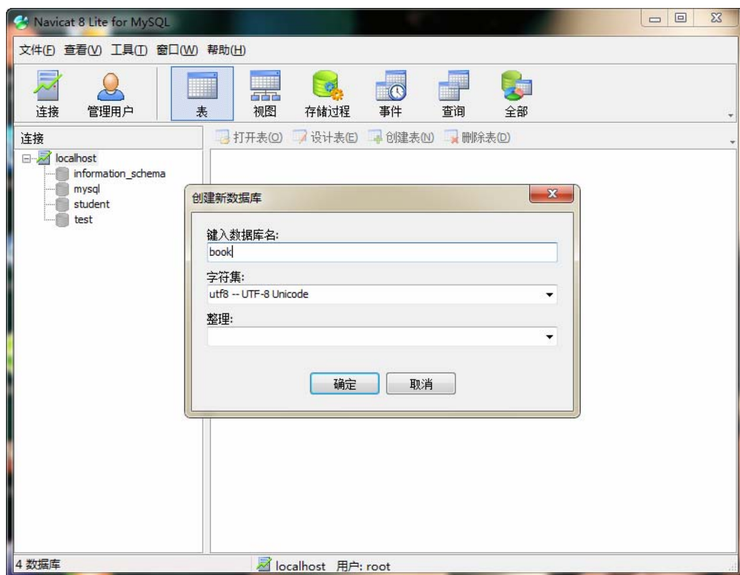


范例图 3-4　执行结果

范例 5　新建 MySQL 数据库

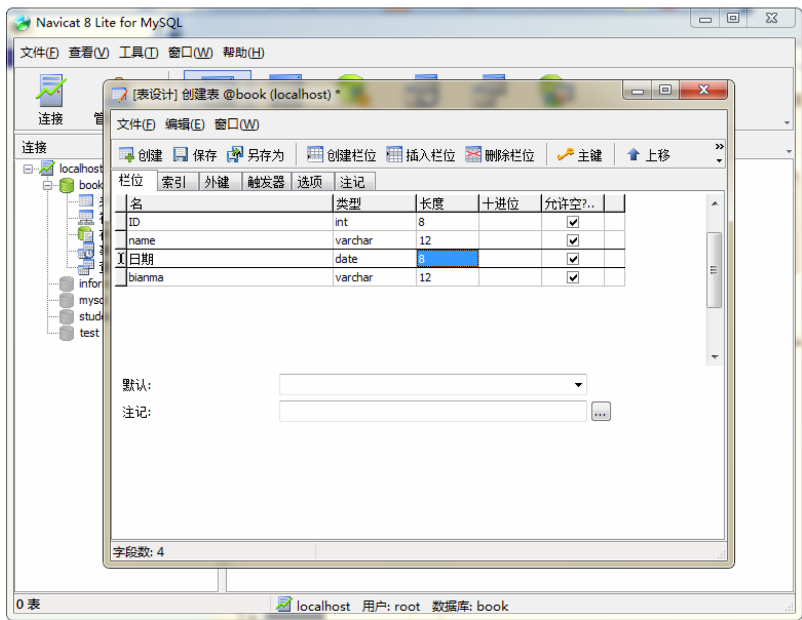
MySQL 是一个优秀的数据库，用户需要熟练地管理和使用它。使用 MySQL 很简单，在前面的章节中已经讲解得很清楚，在这里进行简单回顾，其具体操作如下：

1）启动 Navicat Lite for MySQL 管理软件，在窗口左侧的列表中选中“localhost”选项，单击鼠标右键，在弹出的快捷菜单中选择“新建数据库”命令，打开“创建新数据库”窗口，在“键入数据库名称”文本框中输入“news”，然后单击“确定”按钮，如范例图 3-5 所示。



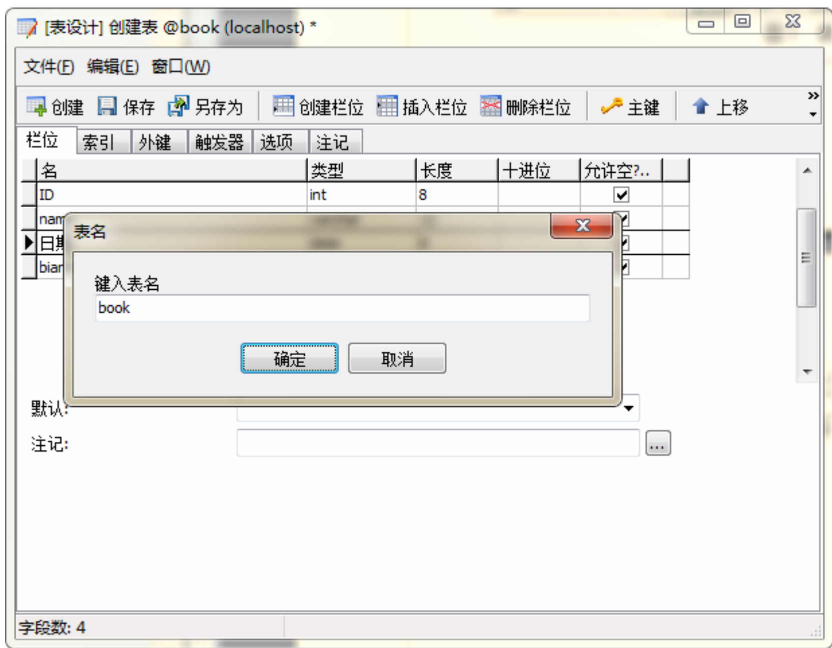
范例图 3-5　新建数据库

2）单击数据库下的表，选中“表”选项，单击鼠标右键，选择“创建表”命令，打开表设计器，然后在名称中输入字段，并设置字段类型和字段的长度，如范例图 3–6 所示。



范例图 3–6　新建表

3）设置完成后，单击“保存”按钮，在弹出的“表名”窗中的文本框中输入表名，这里输入“book”，单击“确定”按钮，如范例图 3–7 所示。



范例图 3–7　输入表名

第四篇　综合实战篇

第 18 章　画 图 板

在 Windows 的附件中有一个出色的画板工具，该工具也是 Microsoft 经典的程序，通过它可以绘制漂亮的图形。当然这个画图工具不是使用 Java 开发的，但是通过 Java 程序也可以实现类似的功能，本章将讲解如何使用 Java 编写画图板程序。

18.1　系统概述与预览

本节将使用 Java 开发一个窗口应用程序，利用该程序可以绘制简单的图画。首先对软件的基本信息和具体功能进行简要介绍。

18.1.1　软件概述

使用 Java 开发窗口程序，肯定会用到 AWT 和 Swing 的知识，它们是窗口的核心，通过使用它们的监听和事件，构成窗口程序。软件界面如图 18-1 所示。

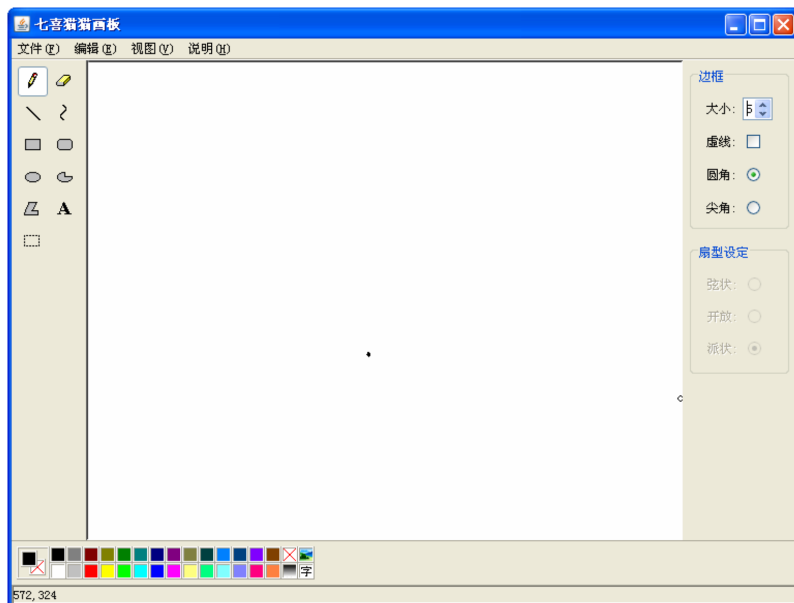


图 18-1　软件界面

18.1.2　软件预览

用户将 Java 源程序复制到一个地方，然后单击“开始”菜单，选择“运行”命令，在里面输入“cmd”命令，用 dos 命令打开目录，如图 18-2 所示。

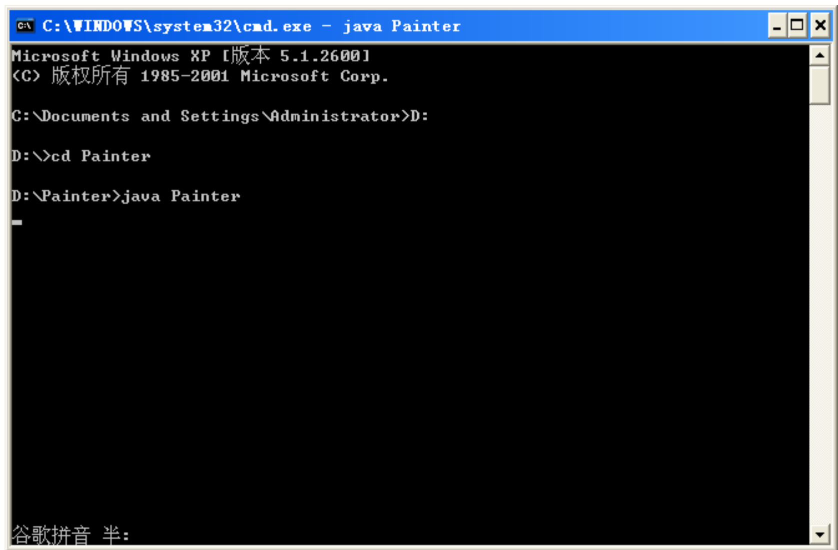


图 18-2　运行画图软件

对于窗口软件来说，菜单尤为重要，如图 18-3 所示为画图软件的菜单命令和快捷键。

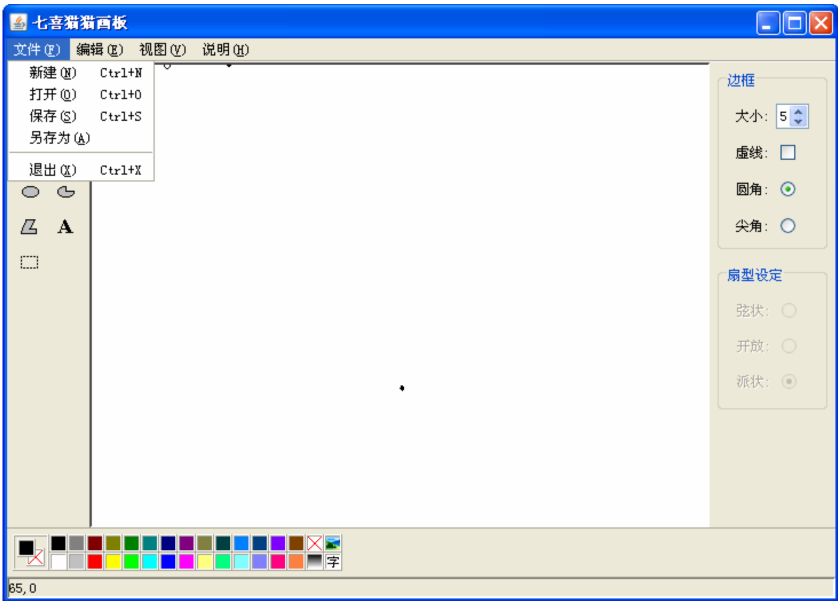
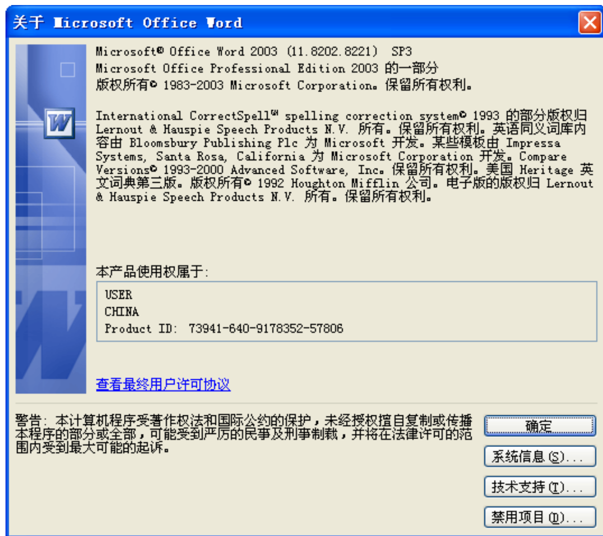


图 18-3　菜单命令

每当制作完成一个软件后，都会有一个对应版权，如 Microsoft 公司的 Word 2003 软件，它的版权如图 18-4 所示。为了让这个软件更加逼真，也给它编写一个关于画板的版

权，如图 18-5 所示。



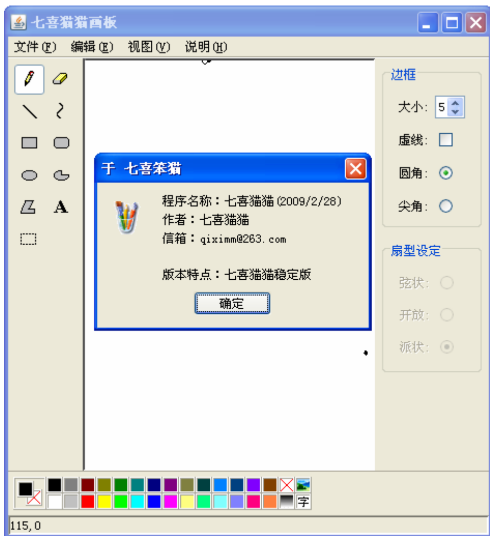


图 18-4　Word 2003 版权　　　　图 18-5　画板版权

对画图软件来说，画图工具是十分重要的，在软件界面的左侧是这个软件的工具栏，如图 18-6 所示。

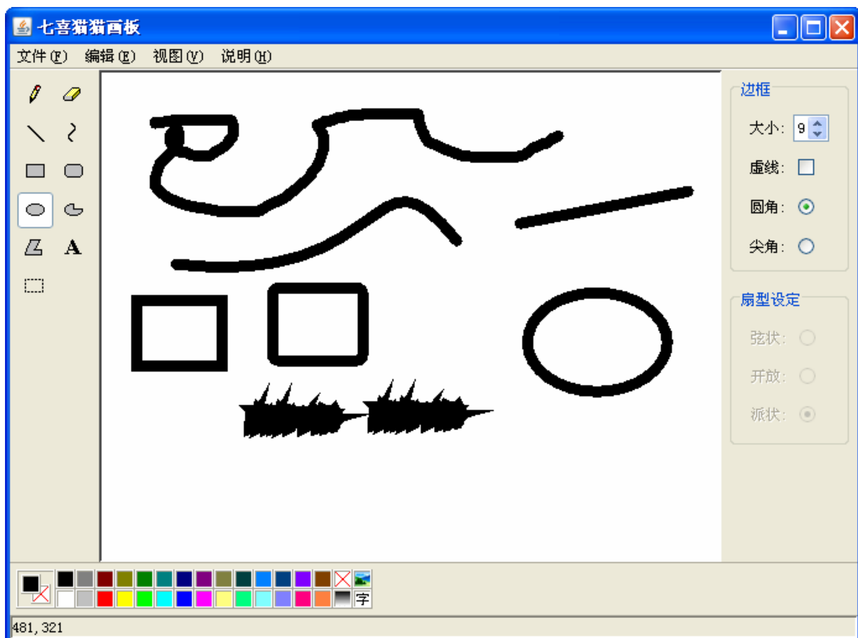


图 18-6　画图软件各种绘图工具

在绘制的图形过程中，丰富的颜色也是十分重要的，这个绘图的软件的颜色工具栏放置在底部，用户需要使用不同的颜色时，只需在下面选择即可，如图 18-7 所示。

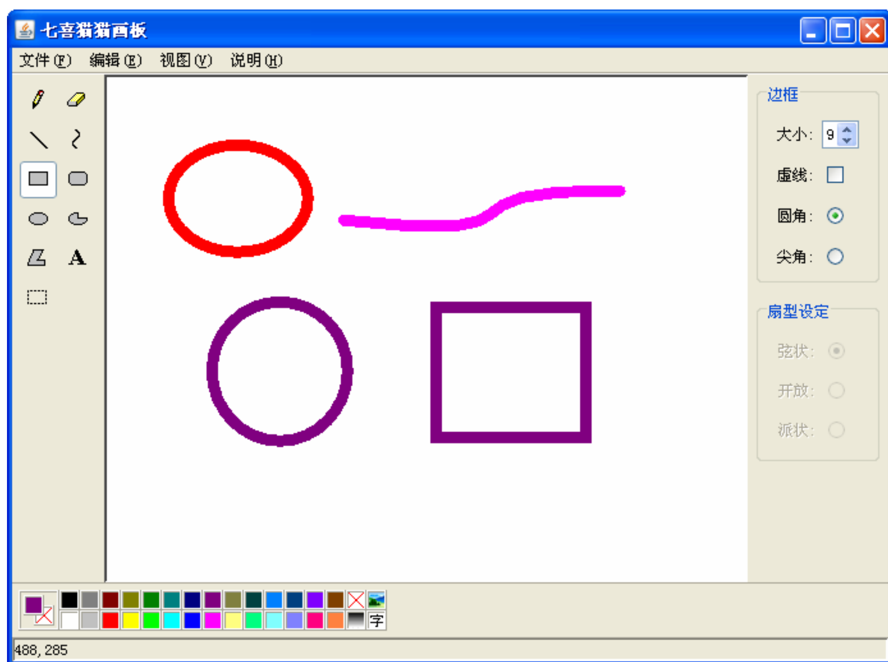


图 18-7　颜色的选择

单使用在软件下部工具栏中的那几种颜色，肯定满足不了用户的需求，用户可以单击如图 18-8 所示的颜色按钮，选择需要的颜色。

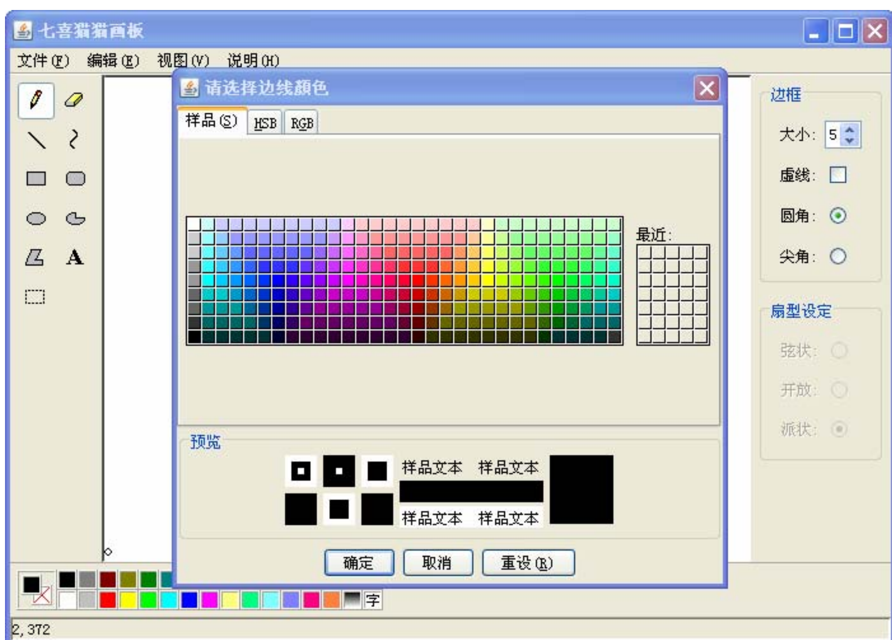


图 18-8　自定义颜色

在软件界面的右侧是一些工具的属性列表，用户可根据需要进行相应设置，如图 18-9 所示。

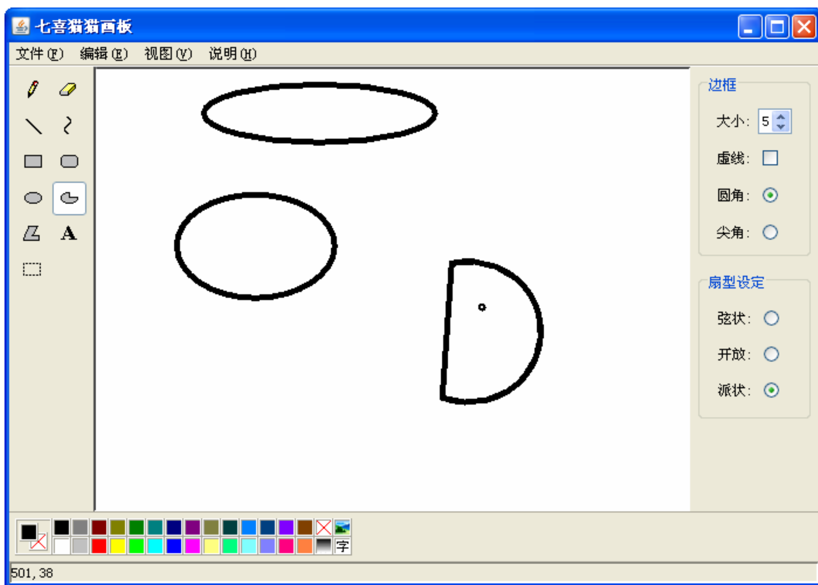


图 18-9　属性设置

18.2　创建软件的准备

在创建这个软件之前，用户不要着急去编写程序，还需要做一些准备工作，下面进行讲解。

18.2.1　搜集素材

通过预览可以发现，在软件界面的左侧和下部的工具栏中有许多图标，这些图标是需要用户自行搜集的，搜集的方法十分简单，用户可以打开 Windows 自带的画板截图后保存即可，如图 18-10 所示。

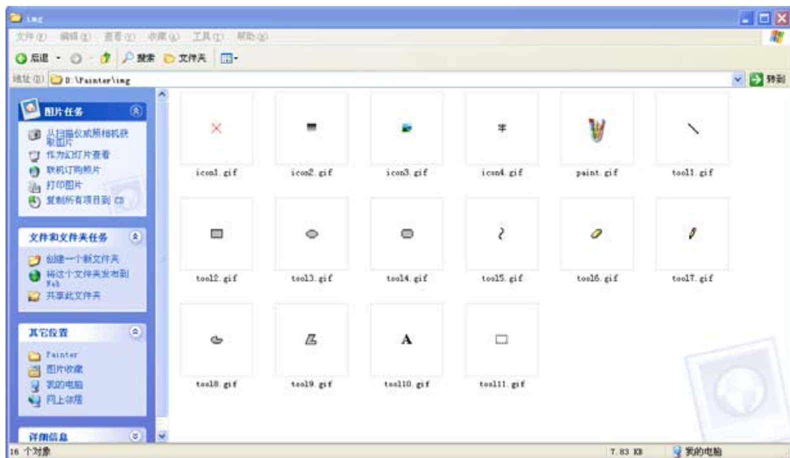


图 18-10　搜集素材

18.2.2 获得 Java API 手册

在编写大型程序软件时，很少有人能记住所有的程序代码，这就需要使用 Java API 手册随时查询需要的方法和类，关于获得 Java API 手册的方法在前面已经讲解过，这里不再赘述，用户可以下载下来或直接在网站上浏览。如图 18−11 所示为网站上的 Java API 手册。



图 18−11　Java API 手册

18.3 编程软件

一个大型软件往往是由无数个小型软件组成的，下面将这个软件分成若干个部分进行讲解。

18.3.1 创建一个类

制作软件的第一步是要创建一个类，让这个类继承 JFrame 并实现 ActionListener 接口，然后估计一下它大概需要插入哪些包，之后编写如下的程序：

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.event.*;
import javax.imageio.ImageIO;
import java.io.*;
import java.awt.geom.*;
import java.awt.image.*;
import java.awt.font.*;
```

```
public class Painter extends JFrame implements ActionListener
 {
}
```

18.3.2　菜单栏和标题栏的程序

在软件界面中，标题栏和菜单栏是尤为重要的，它好比软件的 GPS 导航，引导用户进行操作，下面通过代码进行讲解：

```
private Container c=getContentPane();
   private String menuBar[]={"文件(F)","编辑(E)","视图(V)","说明(H)"};
   private String menuItem[][]={
        {"新建(N)|78","打开(O)|79","保存(S)|83","另存为(A)","退出(X)|88"},
        {"撤销(U)|90","重复(R)|89","剪切(T)|87","复制(C)|68","粘贴(P)|85"},
        {"工具箱(T)|84","色块(C)|76","状态栏(S)","属性栏(M)"},
        {"关于七喜猫猫画板(A)"}
   };
   private JMenuItem jMenuItem[][]=new JMenuItem[4][5];
   private JMenu jMenu[];
   private JCheckBoxMenuItem jCheckBoxMenuItem[]=
new JCheckBoxMenuItem[4];
   private String ButtonName[]=
{"直线","矩形","椭圆","圆角矩形","贝氏曲线","扇形","多边形","铅笔","橡皮擦","
文字","选取"};
   private JToggleButton jToggleButton[];
    private ButtonGroup buttonGroup;
   private JPanel jPanel[]=new JPanel[5];
//1 绘图区,2 工具箱,3 色块,4 属性栏
   private JLabel jLabel[]=new JLabel[1];
//状态列
   private String toolname[]=
{"img/tool1.gif","img/tool2.gif","img/tool3.gif","img/tool4.gif","img/
tool5.gif","img/tool8.gif","img/tool9.gif","img/tool7.gif","img/tool6.gif","im
g/tool10.
gif","img/tool11.gif"};
   private Icon tool[]=new ImageIcon[11];
   private int i,j,show_x,show_y,drawMethod=7,draw_panel_width=700,draw_
panel_height=500;
   private Paint color_border,color_inside;
   private SetPanel setPanel;
   private DrawPanel drawPanel;
   private UnderDrawPanel underDrawPanel;
   private ColorPanel colorPanel;
   private Stroke stroke;
   private Shape shape;
   private String isFilled;
   public Painter()
```

```
        {
                //设定 JMenuBar，并产生 MenuItem、并设置快捷键
                JMenuBar bar=new JMenuBar();
                jMenu=new JMenu[menuBar.length];
                for(i=0;i<menuBar.length;i++){
                     jMenu[i]=new JMenu(menuBar[i]);
                     jMenu[i].setMnemonic(menuBar[i].split("\\(")[1].charAt(0));
                     bar.add(jMenu[i]);
                }

                for(i=0;i<menuItem.length;i++)
        {
                     for(j=0;j<menuItem[i].length;j++)
        {
                        if(i==0 && j==4 || i==1 && j==2) jMenu[i].addSeparator();
                        if(i!=2)
        {
                            jMenuItem[i][j]=new JMenuItem(menuItem[i][j].split ("\\
|")[0]);
                            if(menuItem[i][j].split("\\|").length!=1)

     jMenuItem[i][j].setAccelerator(KeyStroke.getKeyStroke(Integer.parseInt(m
enuItem[i][j].split("\\|")[1]),
        ActionEvent.CTRL_MASK) );
                            jMenuItem[i][j].addActionListener(this);

     jMenuItem[i][j].setMnemonic(menuItem[i][j].split("\\(")[1].charAt(0));
                            jMenu[i].add(jMenuItem[i][j]);
                          }
                          Else
        {
                          jCheckBoxMenuItem[j]=
        new JCheckBoxMenuItem(menuItem[i][j].split("\\|")[0]);
                            if(menuItem[i][j].split("\\|").length!=1)

     jCheckBoxMenuItem[j].setAccelerator(KeyStroke.getKeyStroke(Integer.parse
Int(menuItem[i][j].split("\\|")[1]),
        ActionEvent.CTRL_MASK) );
                            jCheckBoxMenuItem[j].addActionListener(this);
    jCheckBoxMenuItem[j].setMnemonic(menuItem[i][j].split("\\(")[1].charAt(0));
                            jCheckBoxMenuItem[j].setSelected( true );
                            jMenu[i].add(jCheckBoxMenuItem[j]);
                          }
                     }
                }
                this.setJMenuBar( bar );
```

```
        c.setLayout( new BorderLayout() );
        for(i=0;i<5;i++)
             jPanel[i]=new JPanel();
        jLabel[0]=new JLabel(" 状态列");

        buttonGroup=new ButtonGroup();
        JToolBar jToolBar=new JToolBar("工具箱",JToolBar.VERTICAL);
        jToggleButton=new JToggleButton[ButtonName.length];
        for(i=0;i<ButtonName.length;i++)
{
             tool[i] = new ImageIcon(toolname[i]);
             jToggleButton[i]=new JToggleButton(tool[i]);
             jToggleButton[i].addActionListener( this );
             jToggleButton[i].setFocusable( false );
             buttonGroup.add(jToggleButton[i]);
        }
        jToolBar.add(jToggleButton[7]);
        jToolBar.add(jToggleButton[8]);
        jToolBar.add(jToggleButton[0]);
        jToolBar.add(jToggleButton[4]);
        jToolBar.add(jToggleButton[1]);
        jToolBar.add(jToggleButton[3]);
        jToolBar.add(jToggleButton[2]);
        jToolBar.add(jToggleButton[5]);
        jToolBar.add(jToggleButton[6]);
        jToolBar.add(jToggleButton[9]);
        jToolBar.add(jToggleButton[10]);
        jToggleButton[7].setSelected(true);
        jToolBar.setLayout( new GridLayout( 6, 2, 2, 2 ) );
        jPanel[2].add(jToolBar);

        jToolBar.setFloatable(false);
//无法移动

        colorPanel=new ColorPanel();
        jPanel[3].setLayout(new FlowLayout(FlowLayout.LEFT));
        jPanel[3].add(colorPanel);

        drawPanel=new DrawPanel();
        underDrawPanel=new UnderDrawPanel();
        underDrawPanel.setLayout(null);
        underDrawPanel.add(drawPanel);
        drawPanel.setBounds(new Rectangle(2, 2,
draw_panel_width, draw_panel_height));

        setPanel=new SetPanel();
```

```
                jPanel[4].add(setPanel);

                jPanel[0].setLayout( new BorderLayout() );
                jPanel[0].add(underDrawPanel,BorderLayout.CENTER);
                jPanel[0].add(jPanel[2],BorderLayout.WEST);
                jPanel[0].add(jPanel[3],BorderLayout.SOUTH);
                jPanel[0].add(jPanel[4],BorderLayout.EAST);
           jLabel[0].setBorder(BorderFactory.createBevelBorder(BevelBorder.LOW
ERED));
           underDrawPanel.setBorder(BorderFactory.createBevelBorder(BevelBorde
r.LOWERED));
                underDrawPanel.setBackground(new Color(128,128,128));
                jPanel[3].setBorder(BorderFactory.createMatteBorder(1,0,0,0,new
Color(172,168,153)));

                c.add(jPanel[0],BorderLayout.CENTER);
                c.add(jLabel[0],BorderLayout.SOUTH);

                setSize(draw_panel_width,draw_panel_height);
                setTitle("七喜猫猫画板");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                show();
           }
```

18.3.3 保存文档的程序

在编程过程中，经常需要对编程的文件进行保存，下面通过代码讲解保存文件的方法：

```
        public void save()
        {
                FileDialog fileDialog=new FileDialog( new Frame() , "请指定一个
文件名", FileDialog.SAVE );
                fileDialog.show();
                if(fileDialog.getFile()==null) return;
                drawPanel.filename=fileDialog.getDirectory()+fileDialog. getFile
();
           }

           public void actionPerformed( ActionEvent e )
        {
                for(i=0;i<ButtonName.length;i++){
                     if(e.getSource()==jToggleButton[i]){
                          drawMethod=i;
                          if(drawMethod==5)
                               setPanel.pie_add_ctrl();
                          else
```

```
                                setPanel.pie_remove_ctrl();
                           if(drawMethod==7 || drawMethod==8)
                                setPanel.pencil_add_ctrl();
                           else
                                setPanel.pencil_remove_ctrl();
                           drawPanel.clear();
                           drawPanel.repaint();
                               jMenuItem[1][2].setEnabled(false);
                               jMenuItem[1][3].setEnabled(false);
                      }
                 }

                 if(e.getSource()==jMenuItem[1][0])
         {
                      drawPanel.undo();
                 }
                 else if(e.getSource()==jMenuItem[1][1])
         {
                      drawPanel.redo();
                 }
                 else if(e.getSource()==jMenuItem[1][2])
         {
                      drawPanel.cut();
                 }
                 else if(e.getSource()==jMenuItem[1][3])
         {
                      drawPanel.copy();
                 }
                 else if(e.getSource()==jMenuItem[1][4]){
                      drawPanel.paste();
                 }
                 else if(e.getSource()==jMenuItem[0][0])
         {
         //开新文档
                      underDrawPanel.remove(drawPanel);
                      drawPanel=null;
                      drawPanel=new DrawPanel();
                      underDrawPanel.add(drawPanel);
         drawPanel.setBounds(new Rectangle(2, 2, draw_panel_width,
          draw_panel_height));

      underDrawPanel.ctrl_area.setLocation(draw_panel_width+3,draw_panel_heigh
t+3);

      underDrawPanel.ctrl_area2.setLocation(draw_panel_width+3,draw_panel_heig
ht/2+3);
```

```
    underDrawPanel.ctrl_area3.setLocation(draw_panel_width/2+3,draw_panel_he
ight+3);
                repaint();
            }
            else if(e.getSource()==jMenuItem[0][1])
        {
        //开启文档
                FileDialog fileDialog=new FileDialog( new Frame() , "选择
一个文档",
        FileDialog.LOAD );
                fileDialog.show();
                if(fileDialog.getFile()==null) return;

                underDrawPanel.removeAll();
                drawPanel=null;
                drawPanel=new DrawPanel();
                underDrawPanel.add(drawPanel);
               drawPanel.setBounds(new Rectangle(2, 2, draw_panel_width,
         draw_panel_height));

                drawPanel.openfile(fileDialog.getDirectory()+fileDialog.GetFile
());
            }
            else if(e.getSource()==jMenuItem[0][2])
        {
        //存储档案
                if(drawPanel.filename==null)
        {
                    save();
                }
                Else
        {
                    Try
        {
                        int dotpos=drawPanel.filename.lastIndexOf ('.');
                        ImageIO.write(drawPanel.bufImg,
drawPanel.filename.substring(dotpos + 1), new File(drawPanel.filename));
                    }
                    catch(IOException even)
        {
                        JOptionPane.showMessageDialog(null,even.toString
(),"无法存储文档",JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
            else if(e.getSource()==jMenuItem[0][3])
```

```
        {
        //另存新档
                    save();
                    try{
                         int dotpos=drawPanel.filename.lastIndexOf('.');
                         ImageIO.write(drawPanel.bufImg,drawPanel.filename. substring
(dotpos + 1), new File(drawPanel.filename));
                    }
                    catch(IOException even)
         {
                         JOptionPane.showMessageDialog(null,even.toString(),
"无法保存",
         JOptionPane.ERROR_MESSAGE);
                    }
               }
               else if(e.getSource()==jMenuItem[0][4])
        {//离开
                    System.exit(0);
               }
               else if(e.getSource()==jMenuItem[3][0]
        ){//关于
                    JOptionPane.showMessageDialog(null, "程序名称：七喜猫猫
(2009/2/28)\n
        作者：七喜猫猫\n 信箱：qiximm@263.com\n\n 版本特点：七喜猫猫稳定版
        \n", "于 七喜笨猫", 1, new ImageIcon("img/paint.gif"));
               }
               for(i=0;i<2;i++){
                    if(jCheckBoxMenuItem[i].isSelected())
                         jPanel[i+2].setVisible( true );
                    else
                         jPanel[i+2].setVisible( false );
               }
               if(jCheckBoxMenuItem[3].isSelected()){
                    setPanel.setVisible( true );
                    jPanel[4].setVisible( true );
               }
               else{
                    setPanel.setVisible( false );
                    jPanel[4].setVisible( false );
               }
               if(jCheckBoxMenuItem[2].isSelected())
                    jLabel[0].setVisible( true );
               else
                    jLabel[0].setVisible( false );
```

```
        }
```

18.3.4 界面的实现

新建一个类，在这个类中设计界面的布局，然后写入一定的方法，让软件实现大部分功能。下面通过代码讲解实现界面的方法：

```
        public class UnderDrawPanel extends JPanel
        implements MouseListener, MouseMotionListener
        {
                public int x,y;
                float data[]={2};
                public JPanel ctrl_area=new JPanel(),
        ctrl_area2=new JPanel(),ctrl_area3=new JPanel();
                          public UnderDrawPanel()
        {
                     this.setLayout(null);
                     this.add(ctrl_area);
                     this.add(ctrl_area2);
                     this.add(ctrl_area3);
                     ctrl_area.setBounds(new Rectangle(draw_panel_width+3,
         draw_panel_height+3, 5, 5));
                     ctrl_area.setBackground(new Color(0,0,0));
                     ctrl_area2.setBounds(new Rectangle(draw_panel_width+3,
         draw_panel_height/2, 5, 5));
                     ctrl_area2.setBackground(new Color(0,0,0));
                     ctrl_area3.setBounds(new Rectangle(draw_panel_width/2,
         draw_panel_height+3, 5, 5));
                     ctrl_area3.setBackground(new Color(0,0,0));
                     ctrl_area.addMouseListener(this);
                     ctrl_area.addMouseMotionListener(this);
                     ctrl_area2.addMouseListener(this);
                     ctrl_area2.addMouseMotionListener(this);
                     ctrl_area3.addMouseListener(this);
                     ctrl_area3.addMouseMotionListener(this);
                }
                public void mouseClicked(MouseEvent e)
        {
        }
                public void mousePressed(MouseEvent e)
        {
        }
                public void mouseReleased(MouseEvent e)
        {
                     draw_panel_width=x;
                     draw_panel_height=y;
           ctrl_area.setLocation(draw_panel_width+3,draw_panel_height+3);
```

```
     ctrl_area2.setLocation(draw_panel_width+3,draw_panel_height/2+3);
     ctrl_area3.setLocation(draw_panel_width/2+3,draw_panel_height+3);
                drawPanel.setSize(x,y);
                drawPanel.resize();
                repaint();
           }

           public void mouseEntered(MouseEvent e)
         {
         }
                public void mouseExited(MouseEvent e)
         {
         }
                public void mouseDragged(MouseEvent e)
         {
                     if(e.getSource()==ctrl_area2)
         {
                          x=e.getX()+draw_panel_width;
                          y=draw_panel_height;
                     }
                     else if(e.getSource()==ctrl_area3)
         {
                          x=draw_panel_width;
                          y=e.getY()+draw_panel_height;
                     }
                     Else
         {
                          x=e.getX()+draw_panel_width;
                          y=e.getY()+draw_panel_height;
                     }
                     repaint();
                     jLabel[0].setText(x+","+y);
                }
                public void mouseMoved(MouseEvent e)
         {
         }
                     public void paint(Graphics g)
         {
                     Graphics2D g2d=(Graphics2D) g;
                     super.paint(g2d);
                                     g2d.setPaint( new Color(128,128,128) );
                    g2d.setStroke( new BasicStroke( 1,  BasicStroke.CAP_ROUND,
BasicStroke.JOIN_MITER, 10, data, 0 ) );
                     g2d.draw( new Rectangle2D.Double( -1, -1, x+3, y+3 ) );
                }
           }
```

```
                public class SetPanel extends JPanel
        implements ItemListener, ChangeListener, ActionListener{
                private JPanel jPanel_set1=new JPanel();
                private JPanel jPanel_set2=new JPanel();
                private JPanel temp0=new JPanel(new GridLayout(4,1)),
        temp1=new JPanel(new FlowLayout(FlowLayout.LEFT)), temp2=new Jpanel
(new FlowLayout
        (FlowLayout.LEFT)), temp3=new JPanel(new FlowLayout(FlowLayout.LEFT)),
        temp4=new JPanel(new FlowLayout(FlowLayout.LEFT)), temp5=new Jpanel
(new FlowLayout
        (FlowLayout.LEFT)), temp6=new JPanel(new FlowLayout(FlowLayout.LEFT)),
        temp7=new JPanel(new FlowLayout(FlowLayout.LEFT)), temp8=new Jpanel
(new GridLayout
        (3,1));
                public JCheckBox jCheckBox=new JCheckBox();
                private BufferedImage bufImg=
        new BufferedImage(50 ,50,BufferedImage.TYPE_3BYTE_BGR);
                private JLabel jlbImg=new JLabel();
                float data[]={20};
                JLabel pie[]=new JLabel[3];
                public int number=5;
                JSpinner lineWidthSelect=new JSpinner();
                JRadioButton style[]=new JRadioButton[ 5 ];
                ButtonGroup styleGroup=
        new ButtonGroup() ,pieGroup=new ButtonGroup();

                public SetPanel(){//产生版面//
                     this.setLayout(null);
                     this.add(jPanel_set1);

                     jlbImg.setIcon(new ImageIcon(bufImg));
                     jPanel_set1.setLayout(new FlowLayout());
                     jPanel_set1.setBounds(new Rectangle(0, 0, 100, 160));
                     jPanel_set1.setBorder( new TitledBorder(null,
        "边框",TitledBorder.LEFT, TitledBorder.TOP) );
                     lineWidthSelect.setValue(new Integer(5));
                     for(i=0;i<=1;i++)
        {
                          style[i]=new JRadioButton();
                          styleGroup.add(style[i]);
                          style[i].addActionListener(this);
                     }
                     style[0].setSelected( true );
                temp1.add(new JLabel("大小:"));
                     temp1.add(lineWidthSelect);
                     temp2.add(new JLabel("虚线:"));
```

```
                    temp2.add(jCheckBox);

                    temp3.add(new JLabel("圆角:"));
                    temp3.add(style[0]);

                    temp4.add(new JLabel("尖角:"));
                    temp4.add(style[1]);

                    temp0.add(temp1);
                    temp0.add(temp2);
                    temp0.add(temp3);
                    temp0.add(temp4);

                    jPanel_set1.add(temp0);
                    lineWidthSelect.addChangeListener( this );
                    jCheckBox.addItemListener( this );

                    jPanel_set2.setBounds(new Rectangle(0, 170, 100, 130));
                    jPanel_set2.setBorder( new TitledBorder(null, "扇形设定
",TitledBorder.LEFT, TitledBorder.TOP) );

                    for(i=2;i<=4;i++)
        {
                        style[i]=new JRadioButton();
                        pieGroup.add(style[i]);
                        style[i].addActionListener(this);
                    }
                    style[4].setSelected( true );
                    pie[0]=new JLabel("弦状:");
                    temp5.add(pie[0]);
                    temp5.add(style[2]);

                    pie[1]=new JLabel("开放:");
                    temp6.add(pie[1]);
                    temp6.add(style[3]);

                    pie[2]=new JLabel("派状:");
                    temp7.add(pie[2]);
                    temp7.add(style[4]);

                    temp8.add(temp5);
                    temp8.add(temp6);
                    temp8.add(temp7);

                    temp8.setPreferredSize(new Dimension( 71 , 95 ));
```

```
                jPanel_set2.add(temp8);
                this.add(jPanel_set2);

                pie_remove_ctrl();
                stroke=new BasicStroke(5,
BasicStroke.CAP_ROUND, BasicStroke.JOIN_MITER);
        }

        public void pencil_add_ctrl()
{
                style[0].setSelected(true);
                style[1].setEnabled(false);
                jCheckBox.setSelected(false);
                jCheckBox.setEnabled(false);
                BasicStroke stroke2=(BasicStroke) stroke;
                stroke=new BasicStroke(stroke2.getLineWidth(),
 BasicStroke.CAP_ROUND, BasicStroke.JOIN_MITER);
        }

        public void pencil_remove_ctrl()
{
                style[1].setEnabled(true);
                jCheckBox.setEnabled(true);
        }

        public void pie_add_ctrl(){
                pie[0].setEnabled(true);
                pie[1].setEnabled(true);
                pie[2].setEnabled(true);
                style[2].setEnabled(true);
                style[3].setEnabled(true);
                style[4].setEnabled(true);
        }

        public void pie_remove_ctrl()
{
                pie[0].setEnabled(false);
                pie[1].setEnabled(false);
                pie[2].setEnabled(false);
                style[2].setEnabled(false);
                style[3].setEnabled(false);
                style[4].setEnabled(false);
        }

        public void actionPerformed( ActionEvent e )
{
```

```
                    BasicStroke stroke2=(BasicStroke) stroke;
                    if ( e.getSource()==style[0] )
                         stroke=new BasicStroke( stroke2.getLineWidth(),
         BasicStroke.CAP_ROUND, stroke2.getLineJoin(), stroke2.getMiterLimit(),
        stroke2.getDashArray(), stroke2.getDashPhase() );
                    else if ( e.getSource()==style[1] )
                         stroke=new BasicStroke( stroke2.getLineWidth(),
BasicStroke.CAP_BUTT,
         stroke2.getLineJoin(), stroke2.getMiterLimit(),
        stroke2.getDashArray(), stroke2.getDashPhase() );
                    else if ( e.getSource()==style[2] )
                         drawPanel.pie_shape=Arc2D.CHORD;
                    else if ( e.getSource()==style[3] )
                         drawPanel.pie_shape=Arc2D.OPEN;
                    else if ( e.getSource()==style[4] )
                         drawPanel.pie_shape=Arc2D.PIE;
               }

               public void stateChanged(ChangeEvent e)
        {
                    number=Integer.parseInt
        (lineWidthSelect.getValue().toString());
                    if(number <=0)
         {
                         lineWidthSelect.setValue(new Integer(1));
                         number=1;
                    }
                    BasicStroke stroke2=(BasicStroke) stroke;
                    stroke=new BasicStroke( number, stroke2.getEndCap(),
        stroke2.getLineJoin(), stroke2.getMiterLimit(), stroke2.getDashArray(),
        stroke2.getDashPhase() );
               }

               public void itemStateChanged( ItemEvent e )
        {
                    BasicStroke stroke2=(BasicStroke) stroke;
                    if ( e.getSource()==jCheckBox )
        {
                         if ( e.getStateChange()==ItemEvent.SELECTED )
        stroke=new BasicStroke( stroke2.getLineWidth(), stroke2.getEndCap(),
         stroke2.getLineJoin(), 10, data, 0 );
                         else
        stroke=new BasicStroke(stroke2.getLineWidth(), stroke2.getEndCap(),
         stroke2.getLineJoin());
                    }
               }
```

```
            public Dimension getPreferredSize()
        {
                 return new Dimension( 100, 300 );
            }
     }
            public class Gradient extends JPanel{//渐变预览用
            public Color G_color_left=new Color(255,255,255);
            public Color G_color_right=new Color(0,0,0);
            public Gradient()
        {
                 repaint();
            }

            public void paint(Graphics g)
        {
                Graphics2D g2d=(Graphics2D) g;
                 g2d.setPaint(  new  GradientPaint(0,0,G_color_left,100,0,
G_color_right,
         true));
        g2d.fill(new Rectangle2D.Double(0,0,100,25));
            }

            public Dimension getPreferredSize()
        {
                 return new Dimension(100,25);
            }
     }
```

18.3.5　调色盘的实现

新建一个类，通过这个类最低端的调色功能实现调色盘，下面通过代码讲解：

```
     public class ColorPanel extends JPanel
   implements MouseListener,ActionListener
   {//调色盘 class
            private JPanel jPanel_color0[]=new JPanel[5];
            private JPanel jPanel_color1[]=new JPanel[32];
            private JPanel jPanel_color2[]=new JPanel[32];
            private ImageIcon special_color[]= new ImageIcon[4];
            private BufferedImage bufImg=
   new BufferedImage(12 ,12,BufferedImage.TYPE_3BYTE_BGR) ,bufImg2=new
    BufferedImage(12
   ,12,BufferedImage.TYPE_3BYTE_BGR);
            private JLabel jlbImg=new JLabel() ,jlbImg2=new JLabel();
            private ImageIcon icon;
            private JDialog jDialog;
```

```
            private JButton ok, cancel,left,right;
            private Gradient center=new Gradient();

      private int rgb[][]={
                {0,255,128,192,128,255,128,255,0,0,0,0,0,0,128,255,128,
255,0,0,0,128,0,128,128,255,128,255,255,255,255,255},
                {0,255,128,192,0,0,128,255,128,255,128,255,0,0,0,0,
128,255,64,255,128,255, 64,128,0,0,64,128,255,255,255,255},
                {0,255,128,192,0,0,0,0,0,0,128,255,128,255,128,255,64,
128,64,128,255,255,128,255,255,128,0,64,255,255,255,255}
            };

            public ColorPanel()

      {//产生版面
                addMouseListener( this );
                jlbImg.setIcon(new ImageIcon(bufImg));
                jlbImg2.setIcon(new ImageIcon(bufImg2));

                special_color[0]=new ImageIcon( "img/icon1.gif" );
                special_color[1]=new ImageIcon( "img/icon2.gif" );
                special_color[2]=new ImageIcon( "img/icon3.gif" );
                special_color[3]=new ImageIcon( "img/icon4.gif" );

                this.setLayout(null);
                color_border=new Color(0,0,0);
                color_inside=null;

                for(i=0;i<jPanel_color0.length;i++){
                    jPanel_color0[i]=new JPanel();
                    if(i<=2){

    jPanel_color0[i].setBorder(BorderFactory.createEtchedBorder(BevelBorder.
RAISED));
                        jPanel_color0[i].setLayout(null);
                    }
                    else{
                        jPanel_color0[i].setBackground(new
Color(rgb[0][i-3],rgb[1][i-3],rgb[2][i-3]));
                        jPanel_color0[i].setLayout(new GridLayout(1, 1));
                        jPanel_color0[i-2].add(jPanel_color0[i]);
                    }
                }
                for(i=0;i<jPanel_color2.length;i++){
                    jPanel_color2[i]=new JPanel();
```

```
                    jPanel_color2[i].setLayout(new GridLayout(1,1));
                    jPanel_color2[i].setBounds(new Rectangle(2,2,12,12));
                    jPanel_color2[i].setBackground(new Color(rgb[0][i],
rgb[1][i],rgb[2][i]));
                    if(i>=28)
                        jPanel_color2[i].add(new JLabel(special_color
[i-28]));
                }

                for(i=0;i<jPanel_color1.length;i++){
                    jPanel_color1[i]=new JPanel();
                    jPanel_color1[i].setLayout(null);
                    jPanel_color1[i].add(jPanel_color2[i]);
                    this.add(jPanel_color1[i]);
                    if(i%2==0){jPanel_color1[i].setBounds(new Rectangle
(32+i /2*16, 0, 16, 16));}
                    else{jPanel_color1[i].setBounds(new Rectangle(32+i/
2*16, 16, 16, 16));}
        jPanel_color1[i].setBorder(BorderFactory.createEtchedBorder(BevelBorde
r.RAISED));
                }

                jPanel_color0[3].add(jlbImg);
                jPanel_color0[4].add(jlbImg2);

                Graphics2D g2d=bufImg2.createGraphics();
                g2d.setPaint( Color.white );
                g2d.fill( new Rectangle2D.Double(0,0,12,12));
                g2d.setPaint( Color.red );
                g2d.draw( new Line2D.Double(0,0,12,12));
                g2d.draw( new Line2D.Double(11,0,0,11));
                repaint();

                this.add(jPanel_color0[1]);
                this.add(jPanel_color0[2]);
                this.add(jPanel_color0[0]);

                jPanel_color0[0].setBounds(new Rectangle(0,0,32,32));
                jPanel_color0[1].setBounds(new Rectangle(4,4,16,16));
                jPanel_color0[2].setBounds(new Rectangle(12,12,16,16));
                jPanel_color0[3].setBounds(new Rectangle(2,2,12,12));
                jPanel_color0[4].setBounds(new Rectangle(2,2,12,12));

                jDialog = new JDialog(Painter.this, "请选择两种颜色渐变",
true);
            jDialog.getContentPane().setLayout(new FlowLayout());
```

```
                jDialog.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

                jDialog.setSize(250,110);
                JPanel temp=new JPanel(new GridLayout(2,1));
                JPanel up=new JPanel(new FlowLayout());
                JPanel down=new JPanel(new FlowLayout());

                     ok=new JButton("确定");
                     cancel=new JButton("取消");
                     left=new JButton(" ");
                     right=new JButton(" ");
                     center.setBorder(BorderFactory.createEtchedBorder(BevelBorder.
RAISED));
                     up.add(left);
                     up.add(center);
                     up.add(right);
                     down.add(ok);
                     down.add(cancel);
                     temp.add(up);
                     temp.add(down);
                     jDialog.getContentPane().add(temp);

                     ok.addActionListener(this);
                     cancel.addActionListener(this);
                     left.addActionListener(this);
                     right.addActionListener(this);
                }
                public void actionPerformed( ActionEvent e ){
                     if(e.getSource()== left){
                          center.G_color_left=JColorChooser.showDialog( Painter.
this, "请选择边线颜色", center.G_color_left );
                          center.repaint();
                     }
                     else if(e.getSource() == right){
                          center.G_color_right=JColorChooser.showDialog( Painter.
this, "请选择边线颜色", center.G_color_right );
                          center.repaint();
                     }
                     else{
                          jDialog.dispose();
                     }
                }

                public Dimension getPreferredSize()
        {
                     return new Dimension(300,32 );
```

```
                }
                public void mouseClicked( MouseEvent e )
        {
        }
                public void mousePressed( MouseEvent e ){
                     Graphics2D g2d;
                    if(e.getX()>=5 && e.getX()<=20 && e.getY()>=5 && e.getY()<=20)
        {
                          g2d=bufImg.createGraphics();
                          color_border=JColorChooser.showDialog( Painter.
this, "请选择边线颜色", (Color)color_border );
                          g2d.setPaint(color_border);
                          g2d.fill( new Rectangle2D.Double(0,0,12,12));
                          repaint();
                     }
                     else if(e.getX()>=13 && e.getX()<=28 && e.getY()>=13 &&
e.getY()<=28){
                          g2d=bufImg2.createGraphics();
                          color_inside=JColorChooser.showDialog( Painter.
this, "请选择填充颜色",(Color)color_inside );
                          g2d.setPaint(color_inside);
                          g2d.fill( new Rectangle2D.Double(0,0,12,12));
                          repaint();
                     }

                     if(!(e.getX()>=32 && e.getX()<=288)) return;
                     int choose=(e.getX()-32)/16*2+e.getY()/16;

                     if(e.getButton()==1)
        //判断填充边框或填满内部
                          g2d=bufImg.createGraphics();
                     else
                          g2d=bufImg2.createGraphics();

                     if(choose==28){//填充无颜色
                          g2d.setPaint( Color.white );
                          g2d.fill( new Rectangle2D.Double(0,0,12,12));
                          g2d.setPaint( Color.red );
                          g2d.draw( new Line2D.Double(0,0,12,12));
                          g2d.draw( new Line2D.Double(11,0,0,11));
                          repaint();

                          if(e.getButton()==1)
                               color_border=null;
                          else
                               color_inside=null;
```

```
                    }
                    else if(choose==29)
        {//填充渐变
                         jDialog.show();

                         g2d.setPaint(  new  GradientPaint(0,0,center.G_color
_left,12,12,center.G_color_right,true));
                         g2d.fill(new Rectangle2D.Double(0,0,12,12));
                         repaint();

                         if(e.getButton()==1)
                              color_border=new GradientPaint(0,0,center.G_
color_left,12,12,center.G_color_right,true );
                         else
                              color_inside=new GradientPaint(0,0,center.G_
color_left, 12, 12, center.G_color_right, true );
                    }
                    else if(choose==30)
        {//填充图案
                         FileDialog fileDialog=new FileDialog( new Frame() ,
        "选择一个图档", FileDialog.LOAD );//利用 FileDialog 抓取档名
                         fileDialog.show();//秀出视窗
                         if(fileDialog.getFile()==null) return;//按取消的处理
        g2d.drawImage(special_color[2].getImage(), 0, 0,this);
        //把调色盘左方换成『图片』

                         icon=new ImageIcon(fileDialog.getDirectory()
        +fileDialog.getFile());//利用 FileDialog 传进来的档名读取图片
           BufferedImage bufferedImage=new BufferedImage
        (icon.getIconWidth(),icon.getIconHeight(),BufferedImage.TYPE_3BYTE_BGR
);//创一张新的 BufferedImage，为了要读取读进来的图片长宽，以免有空白
           bufferedImage.createGraphics().drawImage(icon.getImage(),0,0,this);
        //把 icon 画到 BufferedImage 上
                         repaint();
        //重绘屏幕
                         if(e.getButton()==1)
        //判断边线颜色或内部填满色
           color_border=new TexturePaint(bufferedImage, new Rectangle
        ( icon.getIconWidth(), icon.getIconHeight() ) );
        //把这张 BufferedImage 设成 TexturePaint 来填满
                         else
                              color_inside=new TexturePaint(bufferedImage,
        new Rectangle( icon.getIconWidth(), icon.getIconHeight() ) );
                    }
                    else if(choose==31)
        {//填充文字
```

```
                            String  text=JOptionPane.showInputDialog("请输入文字","文字");//输入文字
                            if(text==null) return;//按取消时的处理
                      Color FontColor=new Color(0,0,0);//给这个字颜色
        FontColor=JColorChooser.showDialog( Painter.this, "请选择一个颜色当文字颜色", FontColor );//请使用者选择颜色
           g2d.drawImage(special_color[3].getImage(), 0, 0,this);//把调色盘左方换成『字』

        BufferedImage bufferedImage=new BufferedImage
        (draw_panel_width,draw_panel_height,BufferedImage.TYPE_3BYTE_BGR);
        //创一张新的 BufferedImage
           Graphics2D g2d_bufferedImage=bufferedImage.createGraphics();
           FontRenderContext frc=g2d_bufferedImage.getFontRenderContext();
        //读 Graphics 中的 Font
                            Font f=new Font("新细明体",Font.BOLD,10);//新 Font
        TextLayout tl=new TextLayout(text, f, frc);//创新的 TextLayout，并利用f(Font) frc(FontRenderContext)
        int sw=(int) (tl.getBounds().getWidth()+tl.getCharacterCount());//计算TextLayout 的长
        int sh=(int) (tl.getBounds().getHeight()+3);//计算 TextLayout 的高
        bufferedImage=new BufferedImage
        (sw,sh,BufferedImage.TYPE_3BYTE_BGR);
        //再创一张新的 BufferedImage，这里利用相同指标指向不同记忆体
        g2d_bufferedImage=bufferedImage.createGraphics();
        //拿出 Graphics 来画，前一张 BufferedImage 只是为了计算文字长度与高度，这样才能完整填满
        g2d_bufferedImage.setPaint(Color.WHITE);//设定颜色为白色
        g2d_bufferedImage.fill(new Rectangle(0,0,sw,sh));//画一个填满白色矩型
        g2d_bufferedImage.setPaint(FontColor);//设定颜色为之前选择文字颜色
        g2d_bufferedImage.drawString(text,0,10);
        //画一个 String 于 BufferedImage 上
                            repaint();//更新画面

                            if(e.getButton()==1)//判断边线颜色或内部填满色
                                 color_border=new TexturePaint(bufferedImage,
        new Rectangle(sw,sh) );//把这张 BufferedImage 设成 TexturePaint 来填满
                            else
                                 color_inside=new TexturePaint(bufferedImage,
        new Rectangle(sw,sh) );
                      }
                      else{//填充一般色
                            g2d.setPaint(new Color(rgb[0][choose],rgb[1][choose],
        rgb[2][choose]));
                            g2d.fill( new Rectangle2D.Double( 0, 0, 12, 12 ) );
                            repaint();
```

```
                          if(e.getButton()==1)
                               color_border=new
Color(rgb[0][choose],rgb[1][choose],rgb[2][choose]);
                          else
                               color_inside=new
Color(rgb[0][choose],rgb[1][choose],rgb[2][choose]);
                     }
                }

                public void mouseReleased( MouseEvent e )
        {}
                public void mouseEntered( MouseEvent e )
        {}
                public void mouseExited( MouseEvent e )
        {}
           }
```

18.3.6　中央画布的实现

中央画布是绘制图形的区域，若要实现中央画布只需新建一个类，然后写入一些代码即可，下面通过代码讲解：

```
          public class DrawPanel extends JPanel
         implements MouseListener, MouseMotionListener, ItemListener,
        ActionListener, ChangeListener
        {//中央画布
                public BufferedImage bufImg;
        //记录最新画面，并在此上作画
                private BufferedImage bufImg_data[];
        //记录所有画出图面，索引值越大越新，最大为最新
                private BufferedImage bufImg_cut;
                private ImageIcon img;
                private JLabel jlbImg;
                private int x1=-1,y1=-
        1,x2,y2,count,redo_lim,press,temp_x1,temp_y1,temp_x2,temp_y2,temp_x3,t
emp_y3,
        step,step_chk,step_arc,step_chk_arc,chk,first,click,cut;
                private Arc2D.Double arc2D=new Arc2D.Double();
        //扇形
                private Line2D.Double line2D=new Line2D.Double();
        //直线
                private Ellipse2D.Double ellipse2D=new Ellipse2D.Double();
        //椭圆
                private Rectangle2D.Double rectangle2D=new Rectangle2D.
Double();
        //矩形
```

```
                private CubicCurve2D.Double cubicCurve2D=
        new CubicCurve2D.Double();
        //贝氏曲线
                private RoundRectangle2D.Double roundRectangle2D=
        new RoundRectangle2D.Double();
        //圆角矩形
                private Polygon polygon;
        //多边形
                private float data[]={5};
                private Rectangle2D.Double rectangle2D_select=
        new Rectangle2D.Double();//矩形
                private Ellipse2D.Double ellipse2D_pan=new Ellipse2D.
Double();
                private BasicStroke basicStroke_pen=new BasicStroke(1, BasicStroke.
        CAP_ROUND, BasicStroke.JOIN_MITER);
                private BasicStroke basicStroke_select=
        new  BasicStroke(1,  BasicStroke.CAP_ROUND,  BasicStroke.JOIN_MITER,10,
data, 0);
                private double center_point_x;
                private double center_point_y;
                private double start;
                private double end;
                public String filename;
                private JTextField textField_font=
        new JTextField("Fixedsys",16), textField_word=new JTextField("猫猫不累
",16);
                private int size=100;
                private JSpinner fontsize=new JSpinner();
                private JDialog jDialog;
                private JCheckBox bold, italic;
                private JButton ok, cancel;
                public int pie_shape=Arc2D.PIE;
                private int valBold=Font.BOLD;
                private int valItalic=Font.ITALIC;
                private int select_x,select_y,select_w,select_h;

                public void resize()
        {
        //改变大小
                     bufImg=new BufferedImage
        (draw_panel_width, draw_panel_height,BufferedImage.TYPE_3BYTE_BGR);
                     jlbImg=new JLabel(new ImageIcon(bufImg));
        //在 JLabel 上放置 bufImg，用来绘图
                     this.removeAll();
                     this.add(jlbImg);
                     jlbImg.setBounds(new
```

```
        Rectangle(0, 0, draw_panel_width, draw_panel_height));

                    //画出原本图形//
                    Graphics2D g2d_bufImg=(Graphics2D) bufImg.getGraphics();
                    g2d_bufImg.setPaint(Color.white);
                    g2d_bufImg.fill(new
Rectangle2D.Double(0,0,draw_panel_width,draw_panel_height));
                    g2d_bufImg.drawImage(bufImg_data[count],0,0,this);

                    //记录可重做最大次数，并让重做不可按//
                    redo_lim=count++;
                    jMenuItem[1][1].setEnabled(false);

        //新增一张 BufferedImage 型态至 bufImg_data[count]，并将 bufImg 绘制至
bufImg_data[count]//
                    bufImg_data[count]=new BufferedImage
        (draw_panel_width, draw_panel_height, BufferedImage.TYPE_3BYTE_BGR);
                    Graphics2D g2d_bufImg_data=
        (Graphics2D) bufImg_data[count].getGraphics();
                    g2d_bufImg_data.drawImage(bufImg,0,0,this);

                    //判断坐标为新起点//
                    press=0;

                    //复原 MenuItem 可以点选//
                    if(count>0)
                        jMenuItem[1][0].setEnabled(true);
                }

                public DrawPanel() {
                    bufImg_data=new BufferedImage[1000];
                    bufImg=new  BufferedImage(draw_panel_width,draw_panel_height,
BufferedImage.TYPE_3BYTE_BGR);
                    jlbImg=new JLabel(new ImageIcon(bufImg));
        //在 JLabel 上放置 bufImg，用来绘图

                    this.setLayout(null);
                    this.add(jlbImg);
                    jlbImg.setBounds(new Rectangle(0, 0,
        draw_panel_width, draw_panel_height));

                    jMenuItem[1][0].setEnabled(false);
                    jMenuItem[1][1].setEnabled(false);
                    jMenuItem[1][2].setEnabled(false);
                    jMenuItem[1][3].setEnabled(false);
                    jMenuItem[1][4].setEnabled(false);
```

```
                    //画出空白//
                    Graphics2D g2d_bufImg=(Graphics2D) bufImg. getGraphics();
                    g2d_bufImg.setPaint(Color.WHITE);
                    g2d_bufImg.fill(new
Rectangle2D.Double(0,0,draw_panel_width,draw_panel_height));

                    bufImg_data[count]=new BufferedImage(draw_panel_width,
        draw_panel_height, BufferedImage.TYPE_3BYTE_BGR);
                    Graphics2D g2d_bufImg_data=(Graphics2D)
        bufImg_data[count].getGraphics();
                    g2d_bufImg_data.drawImage(bufImg,0,0,this);
```

18.3.7 输入字体的实现

输入字体是该软件必不可少的程序，它具有一些格式，用户可以通过下面代码实现它：

```
                    jDialog=new JDialog(Painter.this, "请选择文字、字形、大小与属
性", true);
                    fontsize.setValue(new Integer(100));
                    bold=new JCheckBox( "粗体" ,true);
                    italic=new JCheckBox( "斜体" ,true);
                    ok = new JButton("确定");
                    cancel=new JButton("取消");
                    JPanel temp_0=new JPanel(new GridLayout(5,1));
                    JPanel temp_1=new JPanel(new FlowLayout(FlowLayout.
LEFT));
                    JPanel temp_2=new JPanel(new FlowLayout(FlowLayout.
LEFT));
                    JPanel temp_3=new JPanel(new FlowLayout(FlowLayout.
LEFT));
                    JPanel temp_4=new JPanel(new FlowLayout());
                    JPanel temp_5=new JPanel(new FlowLayout(FlowLayout.
LEFT));
                    Container jDialog_c=jDialog.getContentPane();
                                   jDialog_c.setLayout(new FlowLayout());
          jDialog.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE );
               jDialog.setSize(250, 200);
               temp_5.add(new JLabel("文字:"));
               temp_5.add(textField_word);
               temp_1.add(new JLabel("字体:"));
               temp_1.add(textField_font);
               temp_2.add(new JLabel("大小:"));
               temp_2.add(fontsize);
               temp_3.add(new JLabel("属性:"));
               temp_3.add(bold);
               temp_3.add(italic);
```

```
            temp_4.add(ok);
            temp_4.add(cancel);
            temp_0.add(temp_5);
            temp_0.add(temp_1);
            temp_0.add(temp_2);
            temp_0.add(temp_3);
            temp_0.add(temp_4);
            jDialog_c.add(temp_0);

            bold.addItemListener(this);
            italic.addItemListener(this);
            fontsize.addChangeListener(this);
            ok.addActionListener(this);
            cancel.addActionListener(this);
            temp_0.setPreferredSize(new Dimension( 180, 150 ));

            repaint();
            addMouseListener(this);
            addMouseMotionListener(this);
        }
            public void stateChanged(ChangeEvent e)
      {
                size=Integer.parseInt(fontsize.getValue().toString());
                if(size <=0)
       {
                    fontsize.setValue(new Integer(1));
                    size=1;
                }
            }

            public void actionPerformed( ActionEvent e )
      {
                jDialog.dispose();
            }
            public void itemStateChanged( ItemEvent e )
      {
                if ( e.getSource() == bold )
                    if ( e.getStateChange() == ItemEvent.SELECTED )
                        valBold=Font.BOLD;
                    else
                        valBold=Font.PLAIN;
                if ( e.getSource() == italic )
                    if ( e.getStateChange() == ItemEvent.SELECTED )
                        valItalic=Font.ITALIC;
                    else
                        valItalic=Font.PLAIN;
```

```
            }

            public Dimension getPreferredSize()
        {
                return new Dimension( draw_panel_width, draw_panel_ height );
            }

            public void openfile(String filename)
        {
```

18.3.8 打开以前文档的实现

如何实现在完成一个图形的绘制或绘制到一半时将其保存后还能再打开这个文件呢？，下面通过代码进行讲解：

```
                Graphics2D g2d_bufImg=(Graphics2D) bufImg.get Graphics();
                ImageIcon icon=new ImageIcon(filename);
                g2d_bufImg.drawImage(icon.getImage(),0,0,this);

                count++;
                bufImg_data[count]=new BufferedImage(draw_panel_width,
         draw_panel_height, BufferedImage.TYPE_3BYTE_BGR);
                Graphics2D g2d_bufImg_data=
        (Graphics2D) bufImg_data[count].getGraphics();
                g2d_bufImg_data.drawImage(bufImg,0,0,this);

                repaint();
            }

            public void undo()
        {
        //复原
                count--;

                draw_panel_width=bufImg_data[count].getWidth();
                draw_panel_height=bufImg_data[count].getHeight();
                drawPanel.setSize(draw_panel_width,draw_panel_height);

                bufImg=new
        BufferedImage(draw_panel_width, draw_panel_height,BufferedImage.TYPE_
3BYTE_BGR);
                jlbImg=new JLabel(new ImageIcon(bufImg));
        //在 JLabel 上放置 bufImg，用来绘图
                this.removeAll();
                this.add(jlbImg);
                jlbImg.setBounds(new Rectangle(0, 0, draw_panel_width,
         draw_panel_height));
```

```
                    Graphics2D g2d_bufImg=(Graphics2D) bufImg.getGraphics();
                    g2d_bufImg.setPaint(Color.white);
                    g2d_bufImg.fill(new
Rectangle2D.Double(0,0,draw_panel_width,draw_panel_height));
                    g2d_bufImg.drawImage(bufImg_data[count],0,0,this);
        underDrawPanel.ctrl_area.setLocation(draw_panel_width+3,draw_panel_hei
ght+3);
        underDrawPanel.ctrl_area2.setLocation(draw_panel_width+3,draw_panel_he
ight/2+3);
        underDrawPanel.ctrl_area3.setLocation(draw_panel_width/2+3,draw_panel_
height+3);

                    underDrawPanel.x=draw_panel_width;
                    underDrawPanel.y=draw_panel_height;

                    if(count<=0)
                         jMenuItem[1][0].setEnabled(false);
               jMenuItem[1][1].setEnabled(true);
               cut=3;
                      repaint();
                  }

               public void redo(){//重做
                    count++;

                    draw_panel_width=bufImg_data[count].getWidth();
                    draw_panel_height=bufImg_data[count].getHeight();
                    drawPanel.setSize(draw_panel_width,draw_panel_height);

                    bufImg=
        new BufferedImage(draw_panel_width,
         draw_panel_height,BufferedImage.TYPE_3BYTE_BGR);
                    jlbImg=new JLabel(new ImageIcon(bufImg));
        //在 JLabel 上放置 bufImg，用来绘图
                    this.removeAll();
                    this.add(jlbImg);
                    jlbImg.setBounds(new Rectangle
        (0, 0, draw_panel_width, draw_panel_height));

                   Graphics2D  g2d_bufImg=(Graphics2D) bufImg.getGraphics ();
                    g2d_bufImg.setPaint(Color.white);
                    g2d_bufImg.fill(new
Rectangle2D.Double(0,0,draw_panel_width,draw_panel_height));
                    g2d_bufImg.drawImage(bufImg_data[count],0,0,this);
        underDrawPanel.ctrl_area.setLocation(draw_panel_width+3,draw_panel_hei
ght+3);
```

```
        underDrawPanel.ctrl_area2.setLocation(draw_panel_width+3,draw_panel_he
ight/2+3);
        underDrawPanel.ctrl_area3.setLocation(draw_panel_width/2+3,draw_panel_
height+3);

                     underDrawPanel.x=draw_panel_width;
                     underDrawPanel.y=draw_panel_height;

                     if(redo_lim<count)
                          jMenuItem[1][1].setEnabled(false);
                     jMenuItem[1][0].setEnabled(true);
                     cut=3;
                        repaint();
                }

                public void cut(){
                     bufImg_cut=new BufferedImage
        ((int)rectangle2D_select.getWidth(),(int)rectangle2D_select.getHeight (),
BufferedImage.TYPE_3BYTE_BGR);
                     BufferedImage copy=bufImg.getSubimage
        ((int)rectangle2D_select.getX(),(int)rectangle2D_select.getY(),(int)
        rectangle2D_select.getWidth(),(int)rectangle2D_select.getHeight());
                     Graphics2D   g2d_bufImg_cut=(Graphics2D)   bufImg_cut.create
Graphics();
                     g2d_bufImg_cut.drawImage(copy,0,0,this);

                     Graphics2D g2d_bufImg=(Graphics2D) bufImg.getGraphics();
                     g2d_bufImg.setPaint(Color.WHITE);
                     g2d_bufImg.fill(new
Rectangle2D.Double((int)rectangle2D_select.getX(),(int)rectangle2D_select.getY
(),(int)
        rectangle2D_select.getWidth(),(int)rectangle2D_select.getHeight()));

                     redo_lim=count++;
                     jMenuItem[1][1].setEnabled(false);

        //新增一张 BufferedImage 形状至 bufImg_data[count]，并将 bufImg 绘制至
bufImg_data[count]//
                     bufImg_data[count]=
        new BufferedImage(draw_panel_width,
        draw_panel_height, BufferedImage.TYPE_3BYTE_BGR);
                     Graphics2D g2d_bufImg_data=
         (Graphics2D) bufImg_data[count].getGraphics();
                     g2d_bufImg_data.drawImage(bufImg,0,0,this);

                     //判断坐标为新起点//
```

```
                press=0;

                //让复原 MenuItem 可以点选//
                  if(count>0)
                        jMenuItem[1][0].setEnabled(true);
                  jMenuItem[1][2].setEnabled(false);
                  jMenuItem[1][3].setEnabled(false);
                jMenuItem[1][4].setEnabled(true);
                cut=3;
                repaint();
            }
            public void copy()
      {
                bufImg_cut=new BufferedImage
      ((int)rectangle2D_select.getWidth(),(int)rectangle2D_select.getHeight (),
      BufferedImage.TYPE_3BYTE_BGR);
                BufferedImage copy=bufImg.getSubimage((int)
      rectangle2D_select.getX(),(int)rectangle2D_select.getY(),(int)
      rectangle2D_select.getWidth(),(int)rectangle2D_select.getHeight());
                Graphics2D  g2d_bufImg_cut=(Graphics2D)  bufImg_cut.create
Graphics();
                g2d_bufImg_cut.drawImage(copy,0,0,this);
                jMenuItem[1][4].setEnabled(true);
                cut=1;
                repaint();
            }
            public void paste()
      {
                cut=2;
                repaint();
            }
            public void mousePressed(MouseEvent e)
       {
                x1=e.getX();
                y1=e.getY();
                if(first==0){
                    polygon=new Polygon();
                    polygon.addPoint(x1, y1);
                    first=1;
                }
                //判断坐标为新起点//
                press=1;
                chk=0;
                if(cut!=2) cut=0;
            }
```

```
            public void mouseReleased(MouseEvent e)
        {
                x2=e.getX();
                y2=e.getY();

                if(step_chk==0)
        //控制贝氏曲线用
                    step=1;
                else if(step_chk==1)
                    step=2;

                if(step_chk_arc==0)
        //控制扇形用
                    chk=step_arc=1;
                else if(step_chk_arc==1)
                    chk=step_arc=2;

                if(drawMethod==6 && click!=1)
        {
                    polygon.addPoint(x2, y2);
                    repaint();
                }
                if(drawMethod==10)
        {
                    if(cut!=2) cut=1;
                    select_x=(int)rectangle2D_select.getX();
                    select_y=(int)rectangle2D_select.getY();
                    select_w=(int)rectangle2D_select.getWidth();
                    select_h=(int)rectangle2D_select.getHeight();
                    jMenuItem[1][2].setEnabled(true);
                    jMenuItem[1][3].setEnabled(true);
                }

                if((step_chk==2 && step==2) || (step_chk_arc==2 && step_arc=
=2) || drawMethod==0 || drawMethod==1 || drawMethod==2 || drawMethod==3 ||
        drawMethod==7 || drawMethod==8 || drawMethod==9 || cut==2)
        {
        //当不是画贝氏曲线或是已经完成贝氏曲线
                    toDraw();
                }
            }
            public void clear()
        {       cut=select_x=select_y=select_w=select_h=step_chk_arc=
        step_arc=first=step_chk=step=0;
                x1=x2=y1=y2=-1;
            }
```

```
            public void toDraw()
      {
                  if(x1<0 || y1<0) return;
      //防止误操作
                  chk=3;
                  draw(x1,y1,x2,y2);
                                    //画出图形至 bufImg//
                  Graphics2D g2d_bufImg=(Graphics2D) bufImg.getGraphics();
                  if(cut!=2){
                        if(color_inside!=null && drawMethod!=8){
                              g2d_bufImg.setPaint(color_inside);
                              g2d_bufImg.fill(shape);
                        }
                        if(color_border!=null && drawMethod!=8){
                              g2d_bufImg.setPaint(color_border);
                              g2d_bufImg.setStroke(stroke);
                              g2d_bufImg.draw(shape);
                        }
                  }
                  else{
                        g2d_bufImg.drawImage(bufImg_cut,x2,y2,this);
                  }
                  repaint();
                  clear();
                  //记录可重做最大次数，并让重做不可按//
                  redo_lim=count++;
                  jMenuItem[1][1].setEnabled(false);

                  //新增一张 BufferedImage 型态至 bufImg_data[count]，并将 bufImg
绘制至 bufImg_data[count]//
                  bufImg_data[count]=new BufferedImage(draw_panel_width,
draw_panel_height, BufferedImage.TYPE_3BYTE_BGR);
                  Graphics2D  g2d_bufImg_data=(Graphics2D)  bufImg_data[count].
getGraphics();
                  g2d_bufImg_data.drawImage(bufImg,0,0,this);

            //判断坐标为新起点//
            press=0;

            //复原 MenuItem 可以点选//
            if(count>0)
                  jMenuItem[1][0].setEnabled(true);
            }

            public void mouseEntered(MouseEvent e){}
```

```
            public void mouseExited(MouseEvent e){}
            public void mouseClicked(MouseEvent e){
                if(click==1){//连点两下时
                    toDraw();
                }
                click=1;
            }

            public void mouseDragged(MouseEvent e){
                x2=e.getX();
                y2=e.getY();
                if(drawMethod==7 || drawMethod==8){
                    draw(x1,y1,x2,y2);
                    x1=e.getX();
                    y1=e.getY();
                }
                if(drawMethod!=9)
                    repaint();
            }

            public void mouseMoved(MouseEvent e) {
                show_x=x2=e.getX();
                show_y=y2=e.getY();

                jLabel[0].setText(show_x+","+show_y);
                click=0;
                if(drawMethod==7 || drawMethod==8 || cut==2)
                    repaint();
            }
```

18.3.9 实现其他功能

除了上述功能外，还有一些功能需要实现，下面通过代码讲解实现这些功能的方法：

```
            public void draw(int input_x1,int input_y1,int input_x2,int input_y2){
                if(drawMethod==0){//直线时，让 shape 为 Line2D
                    shape=line2D;
                    line2D.setLine(input_x1,input_y1,input_x2,input_y2);
                }
                else if(drawMethod==1){//矩形时，让 shape 为 Rectangle2D
                    shape=rectangle2D;

    rectangle2D.setRect(Math.min(input_x1,input_x2),Math.min(input_y1,input_
y2),Math.abs(input_x1-input_x2),Math.abs(input_y1-
        input_y2));
                }
                else if(drawMethod==2)
```

```
        {
        //椭圆时
                        shape=ellipse2D;

     ellipse2D.setFrame(Math.min(input_x1,input_x2),Math.min(input_y1,input_y
2),Math.abs(input_x1-input_x2),Math.abs(input_y1-
        input_y2));
                    }
                    else if(drawMethod==3)
        {
        //圆角矩形
                        shape=roundRectangle2D;
           roundRectangle2D.setRoundRect(Math.min(input_x1,input_x2),Math.min(
input_y1,input_y2),Math.abs(input_x1-input_x2),Math.abs
        (input_y1-input_y2),10.0f,10.0f);
                    }
                    else if(drawMethod==4){//贝氏曲线
                        shape=cubicCurve2D;
                        if(step==0)
        {
           cubicCurve2D.setCurve(input_x1,input_y1,input_x1,input_y1,input_x2,
input_y2,input_x2,input_y2);
                            temp_x1=input_x1;
                            temp_y1=input_y1;
                            temp_x2=input_x2;
                            temp_y2=input_y2;
                            step_chk=0;
                        }
                        else if(step==1){

           cubicCurve2D.setCurve(temp_x1,temp_y1,input_x2,input_y2,input_x2,in
put_y2,temp_x2,temp_y2);
                            temp_x3=input_x2;
                            temp_y3=input_y2;
                            step_chk=1;
                        }
                        else if(step==2)
        {
           cubicCurve2D.setCurve(temp_x1,temp_y1,temp_x3,temp_y3,input_x2,inpu
t_y2,temp_x2,temp_y2);
                            step_chk=2;
                        }
                    }
                    else if(drawMethod==5)
        {
        //扇形，chk 用来防止意外的 repaint//
```

```
                          if(step_arc==0 || chk==1)
        {
        //步骤控制
                               shape=ellipse2D;

          ellipse2D.setFrame(Math.min(input_x1,input_x2),Math.min(input_y1,in
put_y2),Math.abs(input_x1-input_x2),Math.abs
        (input_y1-input_y2));
                               temp_x1=input_x1;
                               temp_y1=input_y1;
                               temp_x2=input_x2;
                               temp_y2=input_y2;
                               step_chk_arc=0;
                          }
                          else if(step_arc==1 || chk==2)
        {
        //步骤控制
                               shape=arc2D;

                               center_point_x=Math.min(temp_x1,temp_x2)+Math.  abs
(temp_x1-temp_x2)/2;
                               center_point_y=Math.min(temp_y1,temp_y2)+Math.abs
(temp_y1-temp_y2)/2;

                               double a=Math.pow(Math.pow(input_x2-center_ point_
x,2)+Math.pow(input_y2-center_point_y,2),0.5);
                               double b = input_x2-center_point_x;
                               if(input_y2>center_point_y)
                                    start=360+Math.acos(b/a)/Math.PI*-180;
                               else
                                    start=Math.acos(b/a)/Math.PI*180;


          arc2D.setArc(Math.min(temp_x1,temp_x2),Math.min(temp_y1,temp_y2),Ma
th.abs(temp_x1-temp_x2),Math.abs(temp_y1-
        temp_y2),start,0,pie_shape);
                               step_chk_arc=1;
                          }
                          else if(step_arc==2 || chk==3)
        {
        //步骤控制
                               shape=arc2D;

          double a=Math.pow(Math.pow(input_x2-center_point_x,2)+Math.pow
        (input_y2-center_point_y,2),0.5);
                               double b=input_x2-center_point_x;
```

```
                                if(input_y2>center_point_y)
                                     end=360+Math.acos(b/a)/Math.PI*-180-start;
                                else
                                     end=Math.acos(b/a)/Math.PI*180-start;
                                if(end<0){end=360-Math.abs(end);}
          arc2D.setArc(Math.min(temp_x1,temp_x2),Math.min(temp_y1,temp_y2),Ma
th.abs(temp_x1-temp_x2),Math.abs(temp_y1-
        temp_y2),start,end,pie_shape);
                                step_chk_arc=2;
                           }
                      }
                      else if(drawMethod==6)
        {
                      //多边形
                           shape=polygon;
                      }
                      else if(drawMethod==7 || drawMethod==8)
        {
                           //任意线&橡皮擦
                           Graphics2D g2d_bufImg=(Graphics2D) bufImg.getGraphics ();

                           shape=line2D;
                           line2D.setLine(input_x1,input_y1,input_x2,input_y2);
                           if(drawMethod==7)
                                g2d_bufImg.setPaint(color_border);
                           else
                                g2d_bufImg.setPaint(Color.white);
                           g2d_bufImg.setStroke(stroke);
                           g2d_bufImg.draw(shape);
                      }

                      else if(drawMethod==9)
        {
                           //文字
                           Graphics2D g2d_bufImg=(Graphics2D) bufImg.getGraphics();
                          FontRenderContext frc=g2d_bufImg.getFontRenderContext ();
                      jDialog.show();

        Font f=new Font(textField_font.getText(),valBold + valItalic,size);
          TextLayout tl=new TextLayout(textField_word.getText(), f, frc);
                      double sw=tl.getBounds().getWidth();
                      double sh=tl.getBounds().getHeight();

                      AffineTransform Tx=AffineTransform.getScaleInstance(1,
1);
                      Tx.translate(input_x2,input_y2+sh);
```

```
                    shape=tl.getOutline(Tx);
               }
               else if(drawMethod==10)
        {
               //选取工具
                         shape=rectangle2D;

     rectangle2D.setRect(Math.min(input_x1,input_x2),Math.min(input_y1,input_
y2),Math.abs(input_x1-input_x2),Math.abs(input_y1-
        input_y2));
                    }
                    if(color_border instanceof GradientPaint)
        {
        //使用渐层填色读取拖拉坐标
                         color_border=new  GradientPaint(  input_x1,input_y1,
(Color)((GradientPaint)color_border).getColor1(), input_x2,input_y2,
        (Color)((GradientPaint)color_border).getColor2(), true );
                    }
                    if(color_inside instanceof GradientPaint){
                         color_inside=new  GradientPaint(  input_x1,input_y1,
(Color)((GradientPaint)color_inside).getColor1(), input_x2,input_y2,
        (Color)((GradientPaint)color_inside).getColor2(), true );
                    }
               }

               public void paint(Graphics g)
         {
                    Graphics2D g2d=(Graphics2D) g;
                    super.paint(g2d);
        //重绘底层 JPanel 以及上面所有组件

                    if(press==1 && drawMethod!=10 && !(x1<0 || y1<0))
        {//绘图在最上面的 JLabel 上，并判断是不是起点才画
                         draw(x1,y1,x2,y2);
                         if(drawMethod==8) return;
                         if(color_inside!=null){
                              g2d.setPaint(color_inside);
                              g2d.fill(shape);
                         }
                         if(color_border!=null){
                              g2d.setPaint(color_border);
                              g2d.setStroke(stroke);
                              g2d.draw(shape);
                         }
                    }
```

```
                    if(drawMethod==10 && cut==0)
        {
        //选取控制、判断是否选取、剪下、或贴上
                         g2d.setPaint(Color.black);
                         g2d.setStroke(basicStroke_select);

     rectangle2D_select.setRect(Math.min(x1,x2),Math.min(y1,y2),Math.abs(x1-
x2),Math.abs(y1-y2));
                         g2d.draw(rectangle2D_select);
                    }
                    if(cut==1)
        {
                         g2d.setPaint(Color.black);
                         g2d.setStroke(basicStroke_select);
                         rectangle2D_select.setRect(select_x,select_y,select
_w,select_h);
                         g2d.draw(rectangle2D_select);
                    }
                    if(cut==2)
        {
                         g2d.drawImage(bufImg_cut,x2,y2,this);
                    }

                    //跟随游标的圆形//
                    if(drawMethod==7 || drawMethod==8){
                         g2d.setPaint(Color.black);
                         g2d.setStroke(basicStroke_pen);

        ellipse2D_pan.setFrame(x2-setPanel.number/2,y2-setPanel.number/2,setPanel.
number,setPanel.number);
                         g2d.draw(ellipse2D_pan);
                    }
               }
          }

          public static void main( String args[] )
        {
          try{UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        }
               catch(Exception e)
        {e.printStackTrace();
        }

               Painter app=new Painter();
               app.setVisible(true);
               app.setExtendedState(Frame.MAXIMIZED_BOTH);
          }
        }
```

第 19 章 “众望书城”网上系统

至此，读者几乎已经学完了所有 Java 的基础知识。所谓的Java SE，读者已尽数领教，接下来将是编写经典程序的时间。本章将讲解并建立一个数据库系统，在调试过程中使用的是 SQL Server 2000 数据库。希望读者认真学习，在学完本章后能够有所领悟。

本章主要内容如下：

- ❑ 数据库设计。
- ❑ 数据库的驱动。
- ❑ 数据库的连接。
- ❑ 数据库的编程。

19.1　效果展示

本实例是通过使用 Java 和 SQL Sever 完成制作的，其中 SQL Sever 用来管理数据，SQL Sever 的操作和 MySQL 大致相同，在前面的章节中已经讲解过，然后通过 Java 设计一个窗口界面，对货物的销售情况进行管理，本节里将展示该系统具备的功能。

运行系统后，用户将会看到如图 19−1 所示的登录界面。



图 19−1　登录界面

输入用户名和密码就可以登录到主界面，该系统是针对多个用户的，不同的用户名对应不同的密码，这在设计比较完整的数据库系统里是至关重要的。制作系统时首先登录界面要醒目，其背景图案也要美观，在程序上还要考虑用的是什么样的网络，是内网还是外网，这里的数据库系统仅应用在本机上，就不用考虑那么多因素了。输入正确的用户名和密码后，将会进入如图 19–2 所示的画面。

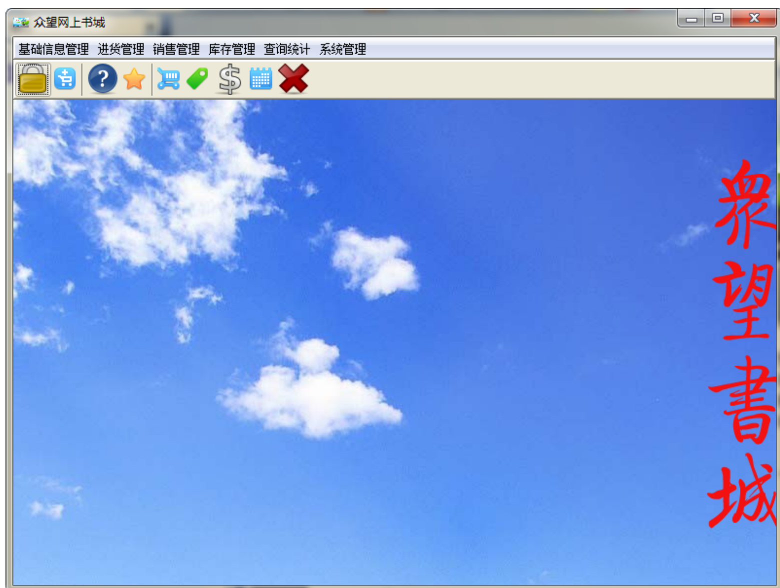


图 19–2 进入管理首页

进入系统后，用户可以选择菜单命令，如选择“查询统计”命令，如图 19–3 所示。



图 19–3 查询设计

该系统作为数据库管理系统，每一个界面都会对数据库进行操作，如图 19–4 所示。

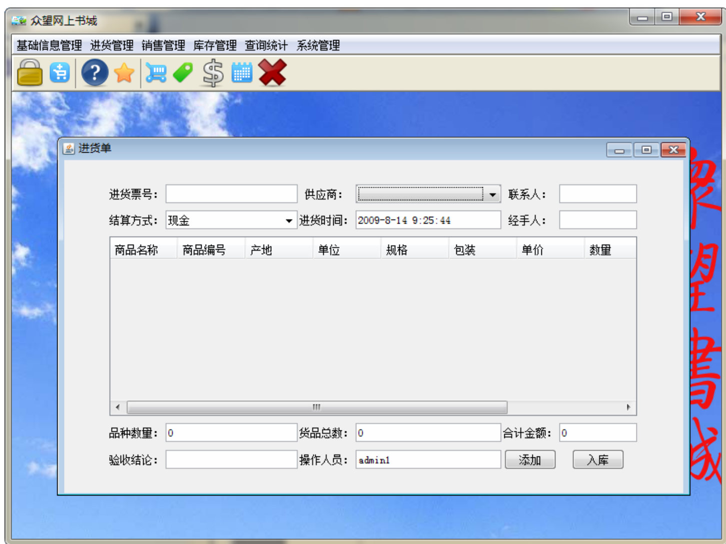


图 19-4　操作数据库

19.2　数据库设计

本书中没有重点讲解数据库设计，但数据库设计是不容小视的，如果它的设计合理，可以为人们的生活提供很多方便。如果它的设计有缺陷，整个系统都会受到限制。本节将展示该系统的数据库设计。

在建立数据库的时候，首先要安装 SQL Sever 数据库，这里以 SQL Sever 2000 为例进行讲解。完成安装后，选择“企业管理器”，创建 db-JXC 数据库，然后创建 tb-userlist 表，如图 19-5 所示，该表是用来管理用户的数据库的。

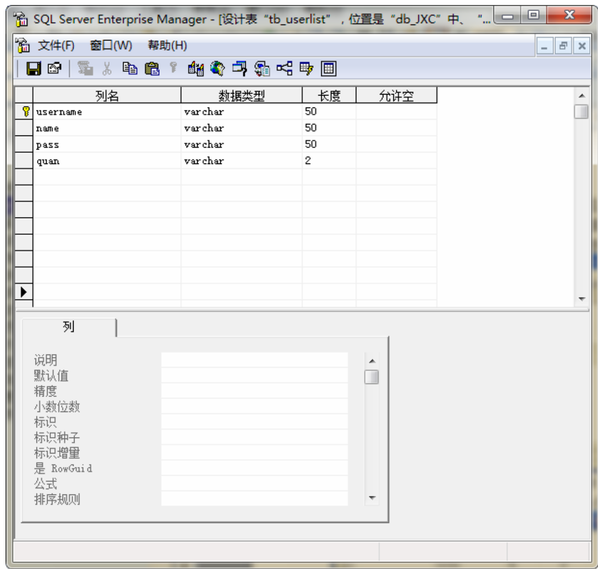


图 19-5　表 tb-userlist

提示：用户如果安装的是 SQL Sever 2000，一定要打上最新的 SP4 补丁，否则无法连接数据库。

接下来创建供应商信息表 tb-gysinfo，如图 19-6 所示，还有商品信息表（tb-spinfo 表），如图 19-7 所示。

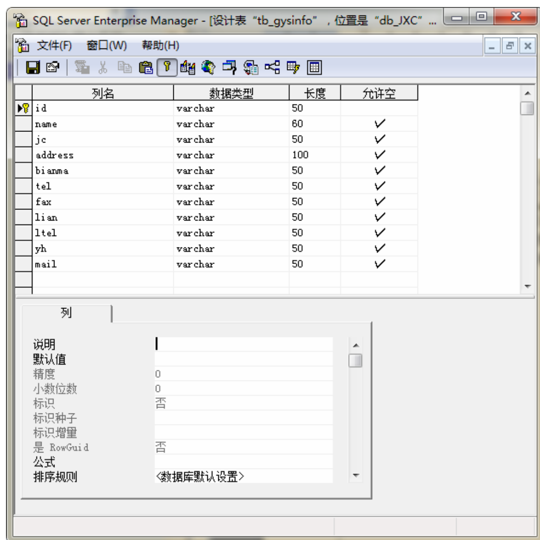


图 19-6　供应商信息表

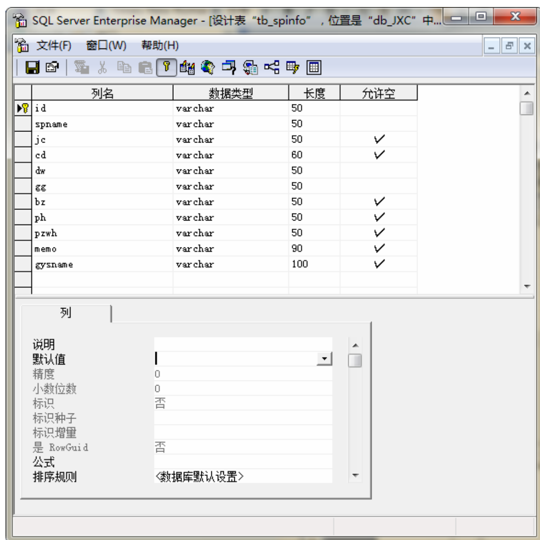


图 19-7　商品信息表

之后创建入库主表 tb-ruku-main，如图 19-8 所示，还有入库明细表 tb-ruku-detail，如图 19-9 所示。

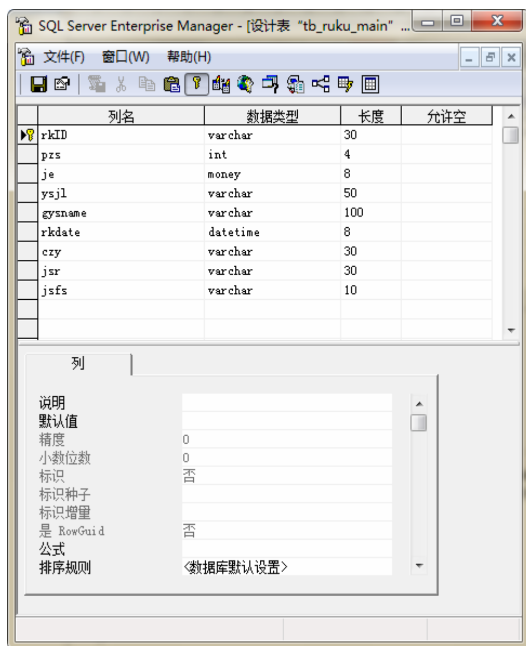


图 19-8　入库主表

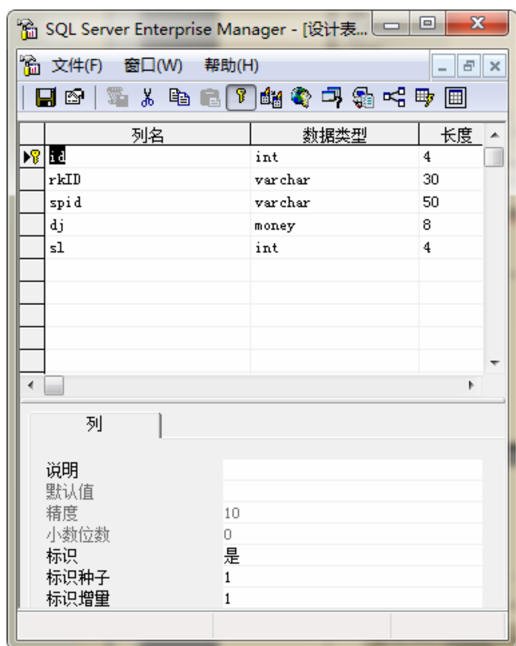


图 19-9　入库明细表

接着创建销售主表 tb-sell-main，如图 19-10 所示。

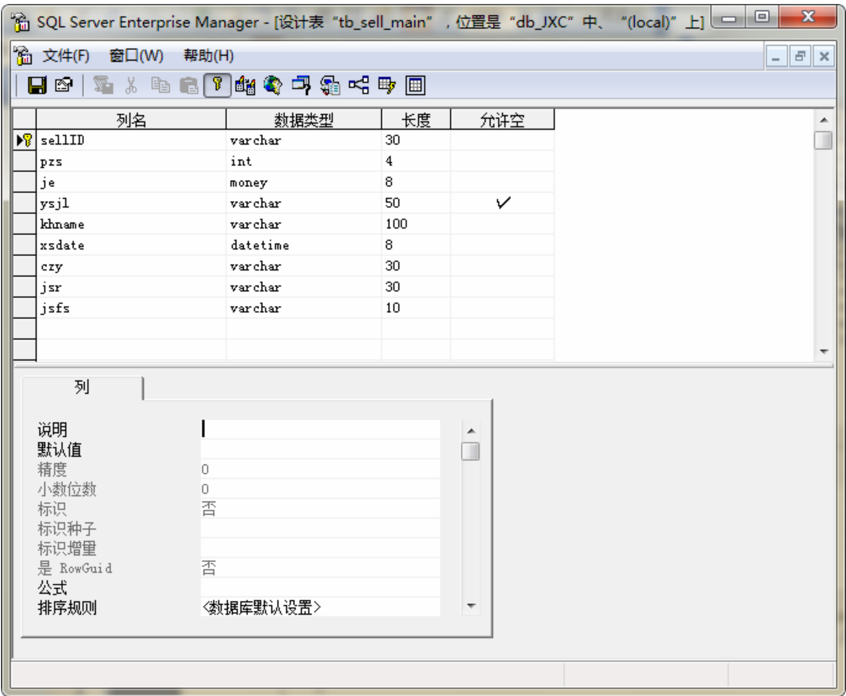


图 19-10 销售主表

还有销售明细表 tb-sell-detail，如图 19-11 所示。

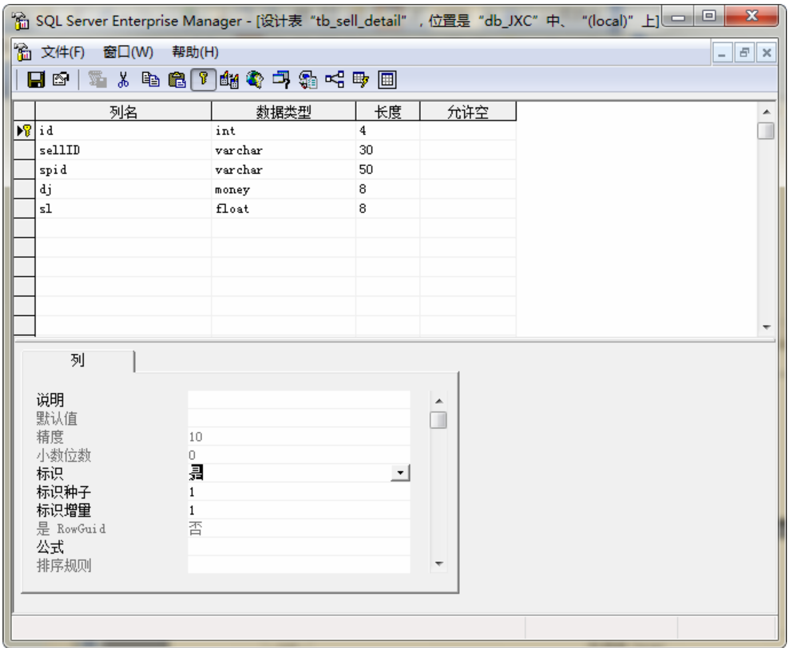
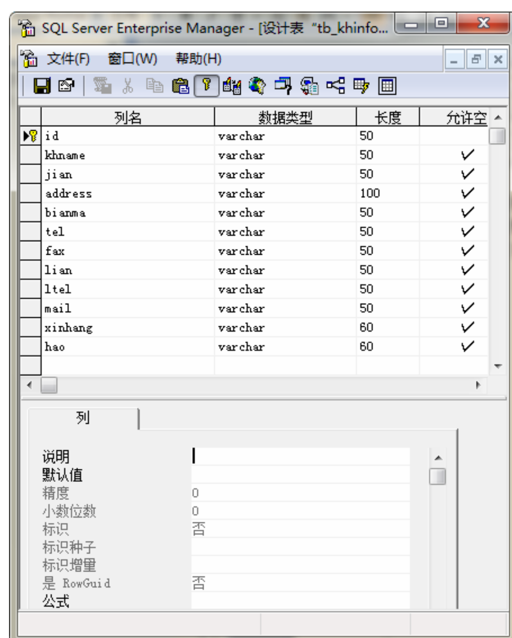


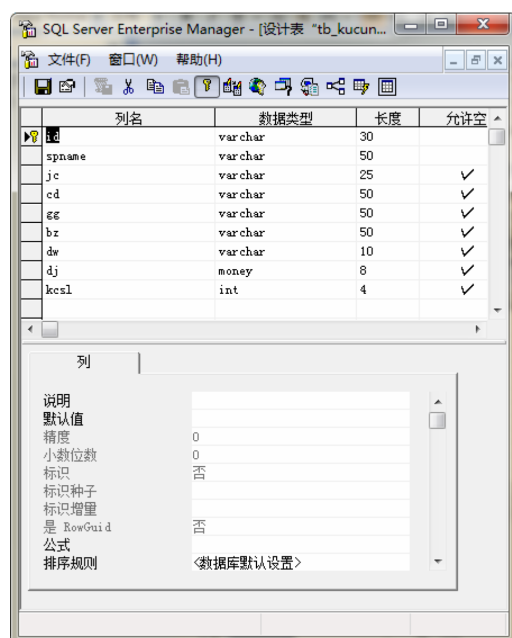
图 19-11 销售明细表

除了上面的表，还需要创建 tb-khinfo 表、tb-kucun 表、tb-xsth-detail 表和 tb-xsth-main 表。如图 19-12 所示为 tb-khinfo 表，如图 19-13 所示为 tb-kucun 表。



列名	数据类型	长度	允许空
id	varchar	50	
khname	varchar	50	✓
jian	varchar	50	✓
address	varchar	100	✓
bianma	varchar	50	✓
tel	varchar	50	✓
fax	varchar	50	✓
lian	varchar	50	✓
ltel	varchar	50	✓
mail	varchar	50	✓
xinhang	varchar	60	✓
hao	varchar	60	✓

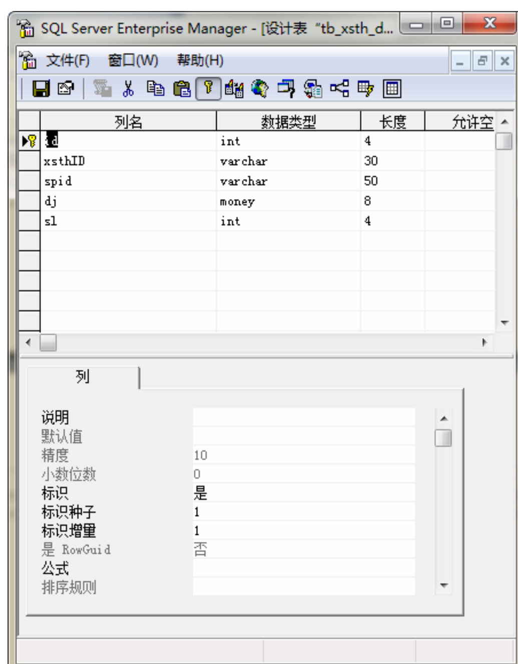
图 19-12　tb-khinfo 表



列名	数据类型	长度	允许空
id	varchar	30	
spname	varchar	50	
jc	varchar	25	✓
cd	varchar	50	✓
gg	varchar	50	✓
bz	varchar	50	✓
dw	varchar	10	✓
dj	money	8	✓
kcsl	int	4	✓

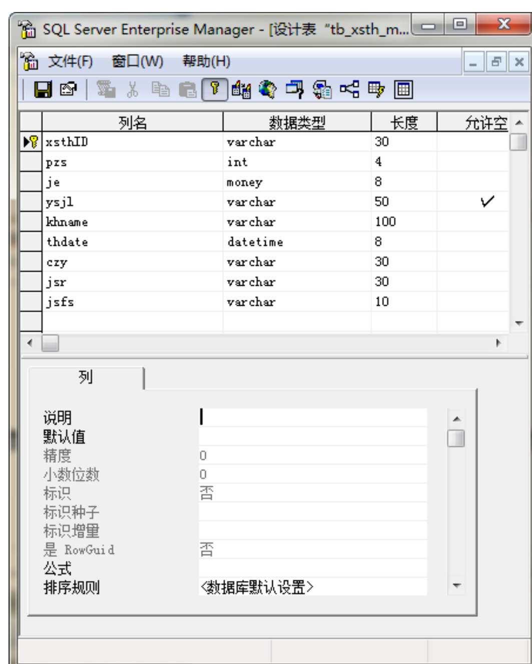
图 19-13　tb-kucun 表

如图 19-14 所示为 tb-xsth-detail 表，如图 19-15 所示为 tb-xsth-main 表。



列名	数据类型	长度	允许空
id	int	4	
xsthID	varchar	30	
spid	varchar	50	
dj	money	8	
sl	int	4	

图 19-14　tb-xsth-detail 表



列名	数据类型	长度	允许空
xsthID	varchar	30	
pzs	int	4	
je	money	8	
ysjl	varchar	50	✓
khname	varchar	100	
thdate	datetime	8	
czy	varchar	30	
jsr	varchar	30	
jsfs	varchar	10	

图 19-15　tb-xsth-main 表

19.3 SQL Server 2000 JDBC 驱动

在一个大型项目中如何寻找驱动？如何下载安装驱动？这些对于经验尚浅的初学者来说有一定的困难，不过没关系只要多加练习，用不了多久就能从菜鸟变成老手。本节将详细讲解设置 JDBC 驱动的方法。

19.3.1 下载 JDBC 驱动

用户可以通过搜索引擎搜索 JDBC 驱动进行下载，这里在 www.gougou.com 网站的搜索栏中输入 sql jdbc sp3，单击“狗狗搜索”按钮将会得到如图 19-16 所示的内容。



图 19-16　搜索 JDBC 驱动

19.3.2 安装 JDBC 驱动

下载完成后，就要对它进行安装，具体安装步骤如下：

1）双击安装文件打开安装程序，单击“Next”按钮，如图 19-17 所示。

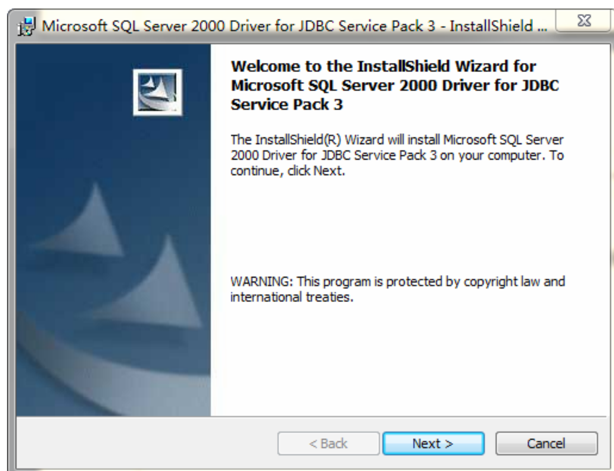


图 19-17　欢迎画面

2）在打开的窗口中选择 I accept the terms in the license agreement 单选按钮，然后单击“Next”按钮，如图 19-18 所示。



图 19-18 遵守协议

3）在打开的窗中选择“Complete”单选按钮，然后单击“Next”按钮，如图 19-19 所示。

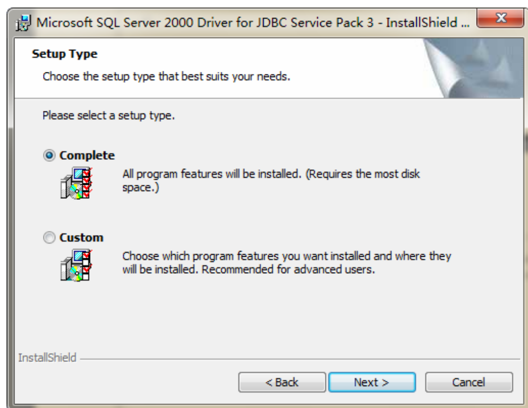


图 19-19 “安装类型”对话框

4）在打开的窗口中单击“Install”按钮开始安装，如图 19-20 所示。

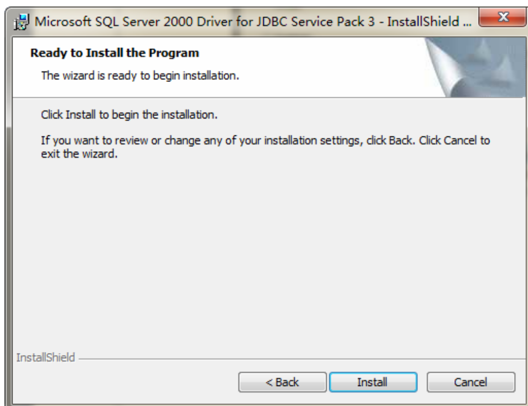


图 19-20 安装

5）单击“Finish”按钮完成安装，如图 19-21 所示。

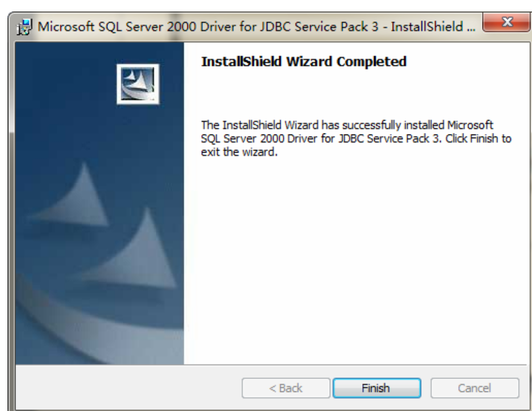


图 19-21　安装完成

6）打开安装目录中的 lib 文件夹（默认路径是 C:\Program Files\Microsoft SQL Server 2000 Driver for JDBC\lib），可见文件夹中有三个 jar 文件，如图 19-22 所示。

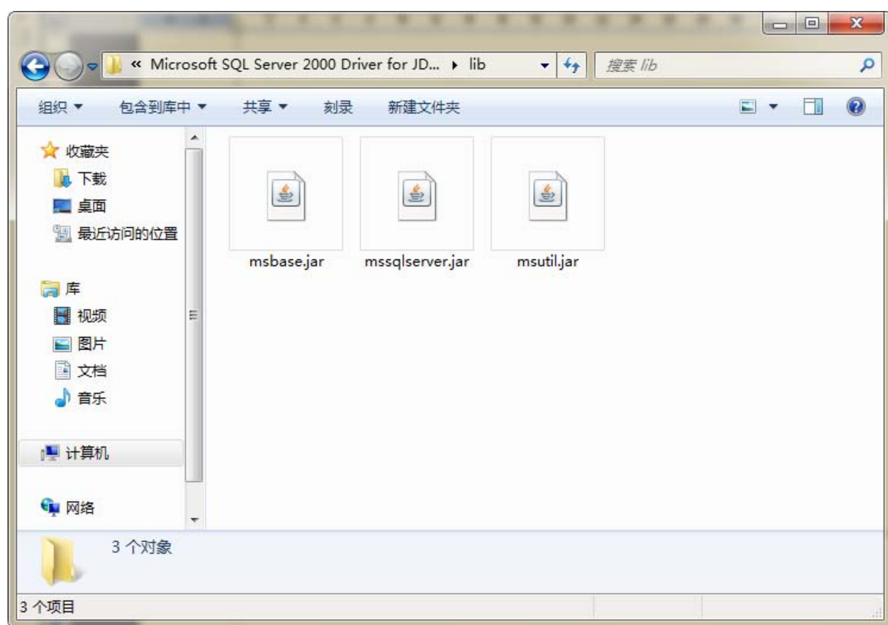


图 19-22　配置文件

提示：安装驱动最主要的目的是获取这三个 jar 文件，这三个文件复制后就算卸载驱动也不会影响数据库的连接，最关键就是这三个文件。

19.3.3　配置 JDBC 驱动

如果用户想在 DOS 环境下也可以进行数据库连接，只需对环境变量进行配置，在变量值文本框中输入上节讲解的三个 jar 文件的路径即可，如图 19-23 所示。

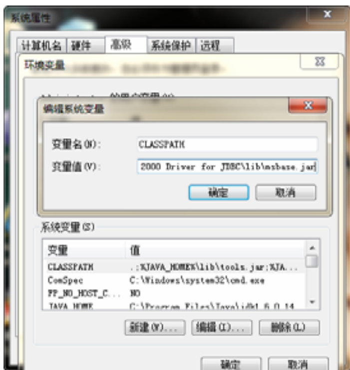


图 19-23　配置 JDBC 驱动

19.3.4　将 JDBC 驱动加载到项目中去

若要将 JDB 驱动加载到项目中，只需将三个 jar 文件加载到 IDE 中即可这里使用的是 Eclipse，其具体操作如下：

1）复制 msbase.jar、mssqlserver.jar 和 msutil.jar 三个文件，然后启动 Eclipse 软件，在左侧列表中项目的分支下单击鼠标右键，选择粘贴命令，如图 19-24 所示。

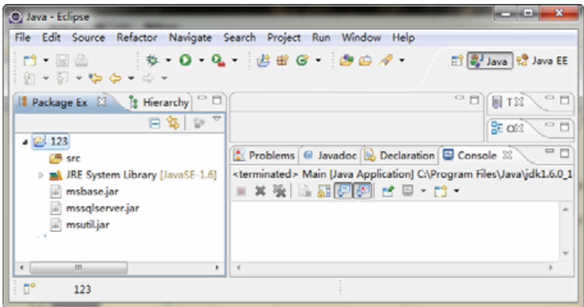


图 19-24　粘贴驱动软件

2）选中一个 jar 文件，单击鼠标右键，选择“Build Path/Add to Build Path”命令，如图 19-25 所示。

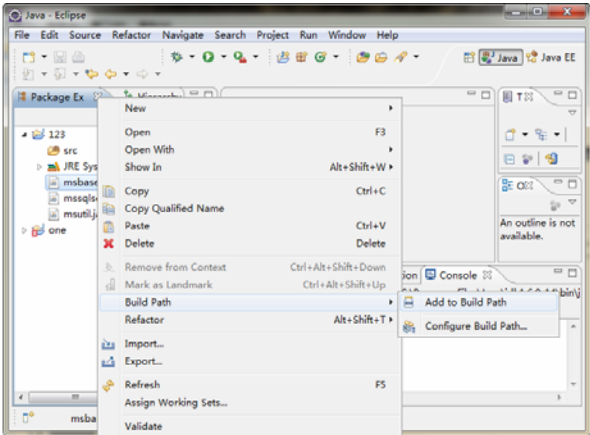


图 19-25　加载驱动

3）用相同的方法加载其他文件，加载完成后如图 19-26 所示。

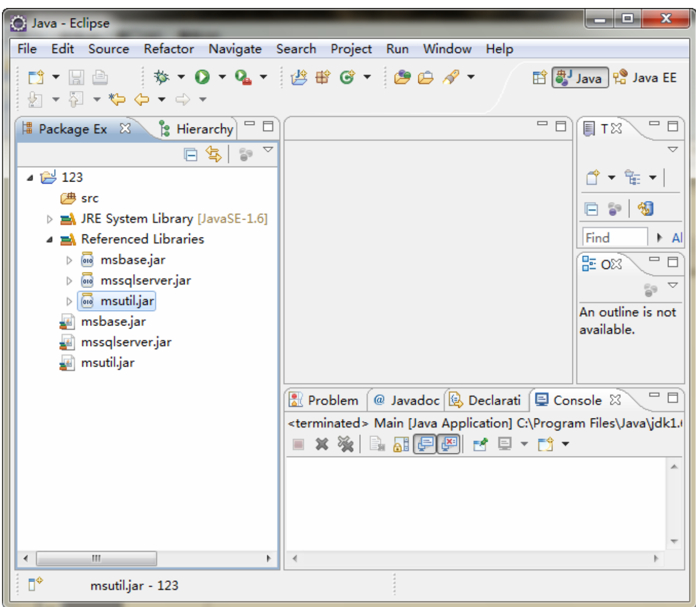


图 19-26　加载驱动

19.4　系统设计

系统设计部分是本章的核心内容，包括登录窗口、主窗口、商品信息的管理、进货信息、销售信息、库存管理、查询与管理等程序。在编程的过程中，程序员需要清楚每一个模块的功能管理、单个模块与数据库的关系，以及模块与模块之间的联系，下面将进行详细讲解。

19.4.1　登录窗口的编写

登录窗口十分简单，就是用户名和密码文本框还有几个简单的按钮，如图 19-27 所示。



图 19-27　登录界面

这个界面与数据库中的表 tb_userlist 息息相关，下面通过代码进行讲解：

```
public class Login extends JFrame
 {
   private JLabel userLabel;
   private JLabel passLabel;
   private JButton exit;
   private JButton login;
   private Main window;
   private static TbUserlist user;
   public Login()
{
         setIconImage(new ImageIcon("res/main1.gif").getImage());
         setTitle("众望书城");
         final JPanel panel=new LoginPanel();
         panel.setLayout(null);
         getContentPane().add(panel);
         setBounds(300,200,panel.getWidth(),panel.getHeight());
         userLabel=new JLabel();
         userLabel.setText("用户名：");
         userLabel.setBounds(140,160,200,18);
         panel.add(userLabel);
         final JTextField userName=new JTextField();
         userName.setBounds(190,160,200,18);
         panel.add(userName);
         passLabel=new JLabel();
         passLabel.setText("密  码：");
         passLabel.setBounds(140,200,200,18);
         panel.add(passLabel);
         final JPasswordField userPassword=new JPasswordField();
         userPassword.addKeyListener(new KeyAdapter()
 {
              public void keyPressed(final KeyEvent e)
{
                   if (e.getKeyCode()==10)
                        login.doClick();
              }
         });
         userPassword.setBounds(190,200,200,18);
         panel.add(userPassword);
         login=new JButton();
         login.addActionListener(new ActionListener()
{
              public void actionPerformed(final ActionEvent e)
 {
                   user=Dao.getUser(userName.getText(), userPassword.
cgetText());
```

```
                    if (user.getUsername()==null || user.getName()==null)
        {
                        userName.setText(null);
                        userPassword.setText(null);
                        return;
                    }
                    setVisible(false);
                    window=new Main();
                    window.frame.setVisible(true);
                }
            });
            login.setText("登录");
            login.setBounds(200,250,60,18);
            panel.add(login);
            exit=new JButton();
            exit.addActionListener(new ActionListener()
        {
                public void actionPerformed(final ActionEvent e)
        {
                    System.exit(0);
                }
            });
            exit.setText("退出");
            exit.setBounds(280,250,60,18);
            panel.add(exit);
            setVisible(true);
            setResizable(false);
            setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);
        }
        public static TbUserlist getUser()
        {
            return user;
        }
        public static void setUser(TbUserlist user)
        {
            Login.user=user;
        }
    }
```

提示：登录界面加载了背景图片，显得更为美观，而且不会影响其他组件的添加，用户可以根据需要设计出更好看的背景图片，告别曾经那个灰白单调的界面。

19.4.2　主窗口

主窗口是用户登录系统后进入的第一个界面，它的设计至关重要，就像人的脸一样，它的好坏将直接影响到用户对整个系统的印象好坏，因此用户一定要做好主界面的设计，

如图 19-28 所示为本系统的主窗口。



图 19-28　主窗口的设计

设计主窗口首先要定义各个组件，并调用登录窗口的运行的程序，下面通过代码进行讲解：

```
        public class Main
        {
           private JDesktopPane desktopPane;
           private JMenuBar menuBar;
           protected JFrame frame;
           private JLabel backLabel;
           // 创建窗体的 Map 类型集合对象
           private  Map<String,  JInternalFrame>  ifs=new  HashMap<String,  JInter
nal Frame>();
           // 创建 Action 动作的 ActionMap 类型集合对象
           private ActionMap actionMap=new ActionMap();
           // 创建并获取当前登录的用户对象
           private TbUserlist user=Login.getUser();
           private Color bgcolor=new Color(Integer.valueOf("ECE9D8", 16));
           public Main()
         {
                 Font font=new Font("宋体", Font.PLAIN, 12);
                 UIManager.put("Menu.font", font);
                 UIManager.put("MenuItem.font", font);
                 // 调用 initialize()方法初始化菜单、工具栏、窗体
                 initialize();
           }
           public static void main(String[] args)
        {
                 SwingUtilities.invokeLater(new Runnable()
```

```
        {
                public void run() {
                    new Login();
                }
            });
    }
    private void initialize()
    {
        frame=new JFrame("众望书屋");
        frame.addComponentListener(new ComponentAdapter()
        {
                public void componentResized(final ComponentEvent e)
        {
                    if (backLabel !=null)
        {
                        int backw=((JFrame) e.getSource()).getWidth();
                        ImageIcon icon=backw <=800 ? new ImageIcon(
                                "res/welcome.jpg") : new ImageIcon(
                                "res/welcomeB.jpg");
                        backLabel.setIcon(icon);
                        backLabel.setSize(backw, frame.getWidth());
                    }
                }
            });
            frame.setIconImage(new ImageIcon("res/main1.gif").getImage());
            frame.getContentPane().setLayout(new BorderLayout());
            frame.setBounds(100,100,800,600);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            desktopPane=new JDesktopPane();
            desktopPane.setBackground(Color.WHITE); // 白色背景
            frame.getContentPane().add(desktopPane);
            backLabel=new JLabel();
            backLabel.setVerticalAlignment(SwingConstants.TOP);
            backLabel.setHorizontalAlignment(SwingConstants.CENTER);
            desktopPane.add(backLabel, new Integer(Integer.MIN_VALUE));
            menuBar=new JMenuBar();
            menuBar.setBounds(0,0,792,66);
            menuBar.setBackground(bgcolor);
            menuBar.setBorder(new LineBorder(Color.BLACK));
            menuBar.setBorder(new BevelBorder(BevelBorder.RAISED));
            frame.setJMenuBar(menuBar);
            menuBar.add(getBasicMenu());              // 添加基础信息菜单
            menuBar.add(getJinHuoMenu());             // 添加进货管理菜单
            menuBar.add(getSellMenu());               // 添加销售管理菜单
            menuBar.add(getKuCunMenu());              // 添加库存管理菜单
            menuBar.add(getCxtjMenu());               // 添加查询统计菜单
```

```
        menuBar.add(getSysMenu());                    // 添加系统设置菜单
        final JToolBar toolBar=new JToolBar("工具栏");
        frame.getContentPane().add(toolBar, BorderLayout.NORTH);
        toolBar.setOpaque(true);
        toolBar.setRollover(true);
        toolBar.setBackground(bgcolor);
        toolBar.setBorder(new BevelBorder(BevelBorder.RAISED));
        defineToolBar(toolBar);
    }
    private JMenu getSysMenu()
   {
  // 获取系统设置菜单
        JMenu menu=new JMenu();
        menu.setText("系统管理");
        JMenuItem item=new JMenuItem();
        item.setAction(actionMap.get("操作员管理"));
        item.setBackground(Color.MAGENTA);
        addFrameAction("操作员管理", "CzyGL", menu);
        addFrameAction("更改密码", "GengGaiMiMa", menu);
        addFrameAction("权限管理", "QuanManager", menu);
        actionMap.put("退出系统", new ExitAction());
        JMenuItem mItem=new JMenuItem(actionMap.get("退出系统"));
        mItem.setBackground(bgcolor);
        menu.add(mItem);
        return menu;
    }
    private JMenu getSellMenu()
   {
    // 获取销售管理菜单
        JMenu menu=new JMenu();
        menu.setText("销售管理");
        addFrameAction("销售单", "XiaoShouDan", menu);
        addFrameAction("销售退货", "XiaoShouTuiHuo", menu);
        return menu;
    }
    private JMenu getCxtjMenu()
  {
  // 获取查询统计菜单
        JMenu menu;
        menu=new JMenu();
        menu.setText("查询统计");
        addFrameAction("客户信息查询", "KeHuChaXun", menu);
        addFrameAction("商品信息查询", "ShangPinChaXun", menu);
        addFrameAction("供应商信息查询", "GongYingShangChaXun", menu);
        addFrameAction("销售信息查询", "XiaoShouChaXun", menu);
        addFrameAction("销售退货查询", "XiaoShouTuiHuoChaXun", menu);
```

```
            addFrameAction("入库查询", "RuKuChaXun", menu);
            addFrameAction("入库退货查询", "RuKuTuiHuoChaXun", menu);
            addFrameAction("销售排行", "XiaoShouPaiHang", menu);
            return menu;
      }
      private JMenu getBasicMenu()
    {
   // 获取基础菜单
            JMenu menu=new JMenu();
            menu.setText("基础信息管理");
            addFrameAction("客户信息管理", "KeHuGuanLi", menu);
            addFrameAction("商品信息管理", "ShangPinGuanLi", menu);
            addFrameAction("供应商信息管理", "GysGuanLi", menu);
            return menu;
      }
      private JMenu getKuCunMenu()
    {
   // 获取库存管理菜单
            JMenu menu=new JMenu();
            menu.setText("库存管理");
            addFrameAction("库存盘点", "KuCunPanDian", menu);
            addFrameAction("价格调整", "JiaGeTiaoZheng", menu);
            return menu;
      }
      private JMenu getJinHuoMenu()
    {
      // 获取进货管理菜单
            JMenu menu=new JMenu();
            menu.setText("进货管理");
            addFrameAction("进货单", "JinHuoDan", menu);
            addFrameAction("进货退货", "JinHuoTuiHuo", menu);
            return menu;
      }
      // 添加工具栏按钮
      private void defineToolBar(final JToolBar toolBar) {
            toolBar.add(getToolButton(actionMap.get("客户信息管理")));
            toolBar.add(getToolButton(actionMap.get("商品信息管理")));
            toolBar.addSeparator();
            toolBar.add(getToolButton(actionMap.get("客户信息查询")));
            toolBar.add(getToolButton(actionMap.get("商品信息查询")));
            toolBar.addSeparator();
            toolBar.add(getToolButton(actionMap.get("库存盘点")));
            toolBar.add(getToolButton(actionMap.get("入库查询")));
            toolBar.add(getToolButton(actionMap.get("价格调整")));
            toolBar.add(getToolButton(actionMap.get("销售单")));
            toolBar.add(getToolButton(actionMap.get("退出系统")));
```

```
        }
        private JButton getToolButton(Action action)
      {
            JButton actionButton=new JButton(action);
            actionButton.setHideActionText(true);
            actionButton.setMargin(new Insets(0, 0, 0, 0));
            actionButton.setBackground(bgcolor);
            return actionButton;
        }
        /*********************辅助方法*************************/
        // 为内部窗体添加 Action 的方法
        private void addFrameAction(String fName, String cname, JMenu menu)
       {
            // System.out.println(fName+".jpg");//输出图片名--调试用
            String img="res/ActionIcon/" + fName + ".png";
            Icon icon=new ImageIcon(img);
            Action action=new openFrameAction(fName, cname, icon);
           if (menu.getText().equals("系统管理") && !fName.equals("更改密码"))
       {
                  if (user==null || user.getQuan()==null
                            || !user.getQuan().equals("a")) {
                        action.setEnabled(false);
                  }
            }
            actionMap.put(fName, action);
            JMenuItem item=new JMenuItem(action);
            item.setBackground(bgcolor);
            menu.add(item);
            if (!menu.getBackground().equals(bgcolor))
                  menu.setBackground(bgcolor);
        }
        // 获取内部窗体的唯一实例对象
        private JInternalFrame getIFrame(String frameName)
       {
            JInternalFrame jf=null;
            if (!ifs.containsKey(frameName))
      {
                  try
      {
                        jf=(JInternalFrame) Class.forName(
                                    "internalFrame." + frameName).get Constructor
(null).newInstance(null);
                        ifs.put(frameName, jf);
                  } catch (Exception e) {
                        e.printStackTrace();
                  }
```

```
                } else
                     jf=ifs.get(frameName);
                return jf;
           }
           // 主窗体菜单项的单击事件监听器
           protected final class openFrameAction extends AbstractAction
         {
                private String frameName=null;
                private openFrameAction()
         {
                }
               public openFrameAction(String cname, String frameName, Icon icon)
         {
                     this.frameName=frameName;
                     putValue(Action.NAME, cname);
                     putValue(Action.SHORT_DESCRIPTION, cname);
                     putValue(Action.SMALL_ICON, icon);
                }
                public void actionPerformed(final ActionEvent e)
        {
                     JInternalFrame jf=getIFrame(frameName);
                     // 在内部窗体闭关时，从内部窗体容器 ifs 对象中清除该窗体。
                     jf.addInternalFrameListener(new InternalFrameAdapter()
        {
                         public void internalFrameClosed(InternalFrameEvent e)
        {
                               ifs.remove(frameName);
                          }
                     });
                     if (jf.getDesktopPane()==null)
        {
                          desktopPane.add(jf);
                          jf.setVisible(true);
                     }
                     try {
                          jf.setSelected(true);
                     } catch (PropertyVetoException e1)
        {
                          e1.printStackTrace();
                     }
                }
           }
           // 退出动作
           protected final class ExitAction extends AbstractAction
        {
                private ExitAction()
```

```
        {
                    putValue(Action.NAME, "退出系统");
                    putValue(Action.SHORT_DESCRIPTION, "众望书城管理系统
1.0");
                    putValue(Action.SMALL_ICON,
                            new ImageIcon("res/ActionIcon/退出系统.png"));
                }
                public void actionPerformed(final ActionEvent e)
        {
                    int exit;
                   exit=JOptionPane.showConfirmDialog(frame.getContent Pane(),
                            "确定要退出吗？", "退出系统", JOptionPane.YES_
NO_OPTION);
                    if (exit==JOptionPane.YES_OPTION)
                        System.exit(0);
                }
            }
            Static
         {
                try
        {
                    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeel
ClassName());
                } catch (Exception e)
        {
                    e.printStackTrace();
                }
            }
        }
```

19.4.3　商品信息的基本管理

在图书城里销售的主要是书，但也不全是书，还可能顺带着销售毛笔、电子琴等等，所以在设计的时候，用户一定要注意不能制作出只能买书的界面。商品信息管理涉及的程序较多，在这里将它分成几部分进行讲解。

（1）商品信息模块的初始化

商品信息管理模块的主要功能是商品信息的添加、修改以及删除，这些功能都是书城系统维护库存的重要信息。商品信息管理模块主要由 JTabbedPane 容纳并初始化，下面通过代码讲解初始化的方法：

```
        public class ShangPinGuanLi extends JInternalFrame
        {
            public ShangPinGuanLi()
        {
                setIconifiable(true);
                setClosable(true);
```

```
                setTitle("商品管理");
                JTabbedPane tabPane=new JTabbedPane();
                final ShangPinXiuGaiPanel spxgPanel=new ShangPinXiuGaiPanel();
                final ShangPinTianJiaPanel sptjPanel=new ShangPinTianJiaPanel();
                tabPane.addTab("商品信息添加", null, sptjPanel, "商品添加");
                tabPane.addTab("商品信息修改与删除", null, spxgPanel, "修改与删除
");
                getContentPane().add(tabPane);
                tabPane.addChangeListener(new ChangeListener()
        {
                     public void stateChanged(ChangeEvent e)
        {
                          spxgPanel.initComboBox();
                          spxgPanel.initGysBox();
                     }
                });
                //在商品管理窗口被激活时，初始化商品添加界面的供应商下拉选择框
                addInternalFrameListener(new InternalFrameAdapter()
        {
                     public void internalFrameActivated(InternalFrameEvent e)
        {
                          super.internalFrameActivated(e);
                          sptjPanel.initGysBox();
                     }
                });
                pack();
                setVisible(true);
           }
        }
```

（2）商品信息的添加

书城中的商品是需要用户自行添加的，该系统添加商品的界面如图 19-29 所示。

图 19-29　商品管理

下面通过代码进行讲解：

```
public class ShangPinTianJiaPanel extends JPanel
{
  private JComboBox gysQuanCheng;
  private JTextField beiZhu;
  private JTextField wenHao;
  private JTextField piHao;
  private JTextField baoZhuang;
  private JTextField guiGe;
  private JTextField danWei;
  private JTextField chanDi;
  private JTextField jianCheng;
  private JTextField quanCheng;
  private JButton resetButton;
  public ShangPinTianJiaPanel()
 {
       setLayout(new GridBagLayout());
       setBounds(10,10,550,400);
       setupComponent(new JLabel("商品名称："),0,0,1,1,false);
       quanCheng=new JTextField();
       setupComponent(quanCheng,1,0,3,1,true);
       setupComponent(new JLabel("简称："),0,1,1,1,false);
       jianCheng=new JTextField();
       setupComponent(jianCheng,1,1,3,10,true);
       setupComponent(new JLabel("产地："),0,2,1,1,false);
       chanDi=new JTextField();
       setupComponent(chanDi,1,2,3,300,true);
       setupComponent(new JLabel("单位："),0,3,1,1,false);
       danWei=new JTextField();
       setupComponent(danWei,1,3,1,130,true);
       setupComponent(new JLabel("规格："),2,3,1,1,false);
       guiGe=new JTextField();
       setupComponent(guiGe,3,3,1,1,true);
       setupComponent(new JLabel("包装："),0,4,1,1,false);
       baoZhuang=new JTextField();
       setupComponent(baoZhuang,1,4,1,1,true);
       setupComponent(new JLabel("批号："),2,4,1,1,false);
       piHao=new JTextField();
       setupComponent(piHao,3,4,1,1,true);
       setupComponent(new JLabel("批准文号："),0,5,1,1,false);
       wenHao=new JTextField();
       setupComponent(wenHao,1,5,3,1,true);
       setupComponent(new JLabel("供应商全称："),0,6,1,1,false);
       gysQuanCheng=new JComboBox();
       gysQuanCheng.setMaximumRowCount(5);
       setupComponent(gysQuanCheng,1,6,3,1,true);
```

```
                setupComponent(new JLabel("备注："),0,7,1,1,false);
                beiZhu=new JTextField();
                setupComponent(beiZhu,1,7,3,1,true);
                final JButton tjButton=new JButton();
                tjButton.addActionListener(new ActionListener()
         {
                     public void actionPerformed(final ActionEvent e)
        {
                          if (baoZhuang.getText().equals("")
                                    || chanDi.getText().equals("")
                                    || danWei.getText().equals("")
                                    || guiGe.getText().equals("")
                                    || jianCheng.getText().equals("")
                                    || piHao.getText().equals("")
                                    || wenHao.getText().equals("")
                                    || quanCheng.getText().equals("")) {
                               JOptionPane.showMessageDialog(ShangPinTianJia
Panel.this,
                               "请完成未填写的信息。", "商品添加", JOptionPane.
ERROR_MESSAGE);
                               return;
                          }
                          ResultSet haveUser=Dao
                                    .query("select * from tb_spinfo where spname='"
                                              + quanCheng.getText().trim() +
"'");
                          try {
                               if (haveUser.next()) {
                                    System.out.println("error");
                                    JOptionPane.showMessageDialog(
                                              ShangPinTianJiaPanel.this, "商
品信息添加失败，存在同名商品",
                                              "客户添加信息", JOptionPane.
INFORMATION_MESSAGE);
                                    return;
                               }
                          } catch (Exception er) {
                               er.printStackTrace();
                          }
                          ResultSet set=Dao.query("select max(id) from tb_spinfo");
                          String id=null;
                          try {
                               if (set !=null && set.next()) {
                                    Stringsid=set.getString(1);
                                    if (sid==null)
                                         id="sp1001";
```

```
                                        else {
                                             String str=sid.substring(2);
                                             id="sp" + (Integer.parseInt(str) + 1);
                                        }
                                   }
                              } catch (SQLException e1) {
                                   e1.printStackTrace();
                              }
                              TbSpinfo spInfo=new TbSpinfo();
                              spInfo.setId(id);
                              spInfo.setBz(baoZhuang.getText().trim());
                              spInfo.setCd(chanDi.getText().trim());
                              spInfo.setDw(danWei.getText().trim());
                              spInfo.setGg(guiGe.getText().trim());
                              spInfo.setGysname(gysQuanCheng.getSelectedItem().
toString()
                                        .trim());
                              spInfo.setJc(jianCheng.getText().trim());
                              spInfo.setMemo(beiZhu.getText().trim());
                              spInfo.setPh(piHao.getText().trim());
                              spInfo.setPzwh(wenHao.getText().trim());
                              spInfo.setSpname(quanCheng.getText().trim());
                              Dao.addSp(spInfo);
                              JOptionPane.showMessageDialog(ShangPinTianJiaPanel.this,
                                        "商品信息已经成功添加", "商品添加",
JOptionPane .INFORMATION_MESSAGE);
                              resetButton.doClick();
                         }
                    });
                    tjButton.setText("添加");
                    setupComponent(tjButton,1,8,1,1,false);
                    final GridBagConstraints gridBagConstraints_20 = new GridBagConst
raints();
                    gridBagConstraints_20.weighty=1.0;
                    gridBagConstraints_20.insets=new Insets(0,65,0,15);
                    gridBagConstraints_20.gridy=8;
                    gridBagConstraints_20.gridx=1;
                    // 重添按钮的事件监听类
                    resetButton=new JButton();
                    setupComponent(tjButton,3,8,1,1,false);
                    resetButton.addActionListener(new ActionListener() {
                         public void actionPerformed(final ActionEvent e) {
                              baoZhuang.setText("");
                              chanDi.setText("");
                              danWei.setText("");
                              guiGe.setText("");
                              jianCheng.setText("");
```

```
                    beiZhu.setText("");
                    piHao.setText("");
                    wenHao.setText("");
                    quanCheng.setText("");
                }
            });
            resetButton.setText("重添");
        }
        // 设置组件位置并添加到容器中
        private void setupComponent(JComponent component, int gridx, int gridy,
                int gridwidth, int ipadx, boolean fill) {
            final  GridBagConstraints  gridBagConstrains=new  GridBagConst
raints();
            gridBagConstrains.gridx=gridx;
            gridBagConstrains.gridy=gridy;
            gridBagConstrains.insets=new Insets(5,1,3,1);
            if (gridwidth > 1)
                gridBagConstrains.gridwidth=gridwidth;
            if (ipadx > 0)
                gridBagConstrains.ipadx=ipadx;
            if (fill)
                gridBagConstrains.fill=GridBagConstraints.HORIZONTAL;
            add(component, gridBagConstrains);
        }
        // 初始化供应商下拉选择框
        public void initGysBox() {
            List gysInfo=Dao.getGysInfos();
            List<Item> items=new ArrayList<Item>();
            gysQuanCheng.removeAllItems();
            for (Iterator iter=gysInfo.iterator(); iter.hasNext();) {
                List element=(List) iter.next();
                Item item=new Item();
                item.setId(element.get(0).toString().trim());
                item.setName(element.get(1).toString().trim());
                if (items.contains(item))
                    continue;
                items.add(item);
                gysQuanCheng.addItem(item);
            }
        }
      }
```

（3）商品信息的编辑

做任何工作都不能保证绝对不出错，当用户输入商品信息出现错误的时候，就必须进行更改或者删除，如图 19-30 所示就是该系统的修改商品信息界面。



图 19-30　修改商品信息

商品信息的编辑主要是负责修改错误的商品信息或者删除不再使用的商品信息，其代码如下：

```
public class ShangPinXiuGaiPanel extends JPanel {
   private JComboBox gysQuanCheng;
   private JTextField beiZhu;
   private JTextField wenHao;
   private JTextField piHao;
   private JTextField baoZhuang;
   private JTextField guiGe;
   private JTextField danWei;
   private JTextField chanDi;
   private JTextField jianCheng;
   private JTextField quanCheng;
   private JButton modifyButton;
   private JButton delButton;
   private JComboBox sp;
   public ShangPinXiuGaiPanel() {
         setLayout(new GridBagLayout());
         setBounds(10, 10, 550, 400);

         setupComponet(new JLabel("商品名称："),0,0,1,1,false);
         quanCheng=new JTextField();
         quanCheng.setEditable(false);
         setupComponet(quanCheng,1,0,3,1,true);

         setupComponet(new JLabel("简称："),0,1,1,1,false);
         jianCheng=new JTextField();
         setupComponet(jianCheng,1,1,3,10,true);

         setupComponet(new JLabel("产地："),0,2,1,1,false);
```

```
chanDi=new JTextField();
setupComponet(chanDi,1,2,3,300,true);

setupComponet(new JLabel("单位："),0,3,1,1,false);
danWei=new JTextField();
setupComponet(danWei,1,3,1,130,true);

setupComponet(new JLabel("规格："),2,3,1,1,false);
guiGe=new JTextField();
setupComponet(guiGe,3,3,1,1,true);

setupComponet(new JLabel("包装："),0,4,1,1,false);
baoZhuang=new JTextField();
setupComponet(baoZhuang,1,4,1,1,true);

setupComponet(new JLabel("批号："),2,4,1,1,false);
piHao=new JTextField();
setupComponet(piHao,3,4,1,1,true);

setupComponet(new JLabel("批准文号："),0,5,1,1,false);
wenHao=new JTextField();
setupComponet(wenHao,1,5,3,1,true);

setupComponet(new JLabel("供应商全称："),0,6,1,1,false);
gysQuanCheng=new JComboBox();
gysQuanCheng.setMaximumRowCount(5);
setupComponet(gysQuanCheng,1,6,3,1,true);

setupComponet(new JLabel("备注："),0,7,1,1,false);
beiZhu=new JTextField();
setupComponet(beiZhu,1,7,3,1,true);

setupComponet(new JLabel("选择商品"),0,8,1,0,false);
sp=new JComboBox();
sp.setPreferredSize(new Dimension(230,21));
// 处理客户信息的下拉选择框的选择事件
sp.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
          doSpSelectAction();
     }
});
// 定位商品信息的下拉选择框
setupComponet(sp,1,8,2,0,true);
modifyButton=new JButton("修改");
delButton=new JButton("删除");
JPanel panel=new JPanel();
panel.add(modifyButton);
panel.add(delButton);
```

```
				// 定位按钮
				setupComponet(panel,3,8,1,0,false);
				// 处理删除按钮的单击事件
				delButton.addActionListener(new ActionListener() {
					public void actionPerformed(ActionEvent e) {
						Item item=(Item) sp.getSelectedItem();
						if (item==null || !(item instanceof Item))
							return;
						int confirm=JOptionPane.showConfirmDialog(
								ShangPinXiuGaiPanel.this, "确认删除商品信息
吗？");
						if (confirm==JOptionPane.YES_OPTION) {
							int rs=Dao.delete("delete tb_spinfo where id='"
									+ item.getId() + "'");
							if (rs > 0) {
								JOptionPane.showMessageDialog(ShangPin
XiuGaiPanel.this,
										"商品：" + item.getName() + "。
删除成功");
								sp.removeItem(item);
							}
						}
					}
				});
				// 处理修改按钮的单击事件
				modifyButton.addActionListener(new ActionListener() {
					public void actionPerformed(ActionEvent e) {
						Item item=(Item) sp.getSelectedItem();
						TbSpinfo spInfo=new TbSpinfo();
						spInfo.setId(item.getId());
						spInfo.setBz(baoZhuang.getText().trim());
						spInfo.setCd(chanDi.getText().trim());
						spInfo.setDw(danWei.getText().trim());
						spInfo.setGg(guiGe.getText().trim());
						spInfo.setGysname(gysQuanCheng.getSelectedItem().toString()
								.trim());
						spInfo.setJc(jianCheng.getText().trim());
						spInfo.setMemo(beiZhu.getText().trim());
						spInfo.setPh(piHao.getText().trim());
						spInfo.setPzwh(wenHao.getText().trim());
						spInfo.setSpname(quanCheng.getText().trim());
						if (Dao.updateSp(spInfo)==1)
							JOptionPane.showMessageDialog(ShangPin
XiuGaiPanel.this,
									"修改完成");
						else
							JOptionPane.showMessageDialog(ShangPinXiuGaiPanel.
this,
```

```
                                        "修改失败");
                    }
                });
        }
        // 初始化商品下拉选择框
        public void initComboBox() {
                List khInfo=Dao.getSpInfos();
                List<Item> items=new ArrayList<Item>();
                sp.removeAllItems();
                for (Iterator iter=khInfo.iterator(); iter.hasNext();) {
                        List element=(List) iter.next();
                        Item item=new Item();
                        item.setId(element.get(0).toString().trim());
                        item.setName(element.get(1).toString().trim());
                        if (items.contains(item))
                                continue;
                        items.add(item);
                        sp.addItem(item);
                }
                doSpSelectAction();
        }
        // 初始化供应商下拉选择框
        public void initGysBox() {
                List gysInfo=Dao.getGysInfos();
                List<Item> items=new ArrayList<Item>();
                gysQuanCheng.removeAllItems();
                for (Iterator iter=gysInfo.iterator(); iter.hasNext();) {
                        List element=(List) iter.next();
                        Item item=new Item();
                        item.setId(element.get(0).toString().trim());
                        item.setName(element.get(1).toString().trim());
                        if (items.contains(item))
                                continue;
                        items.add(item);
                        gysQuanCheng.addItem(item);
                }
                doSpSelectAction();
        }
        // 设置组件位置并添加到容器中
        private void setupComponet(JComponent component, int gridx, int gridy,
                        int gridwidth, int ipadx, boolean fill) {
                final GridBagConstraints gridBagConstrains=new GridBagConst
raints();
                gridBagConstrains.gridx=gridx;
                gridBagConstrains.gridy=gridy;
                if (gridwidth>1)
                        gridBagConstrains.gridwidth=gridwidth;
                if (ipadx>0)
```

```
                gridBagConstrains.ipadx=ipadx;
            gridBagConstrains.insets=new Insets(5,1,3,1);
            if (fill)
                gridBagConstrains.fill=GridBagConstraints.HORIZONTAL;
            add(component, gridBagConstrains);
        }
        // 处理商品选择事件
        private void doSpSelectAction() {
            Item selectedItem;
            if (!(sp.getSelectedItem() instanceof Item)) {
                return;
            }
            selectedItem=(Item) sp.getSelectedItem();
            TbSpinfo spInfo=Dao.getSpInfo(selectedItem);
            if (!spInfo.getId().isEmpty()) {
                quanCheng.setText(spInfo.getSpname());
                baoZhuang.setText(spInfo.getBz());
                chanDi.setText(spInfo.getCd());
                danWei.setText(spInfo.getDw());
                guiGe.setText(spInfo.getGg());
                jianCheng.setText(spInfo.getJc());
                beiZhu.setText(spInfo.getMemo());
                piHao.setText(spInfo.getPh());
                wenHao.setText(spInfo.getPzwh());
                beiZhu.setText(spInfo.getMemo());
                // 设置供应商下拉框的当前选择项
                Item item=new Item();
                item.setId(null);
                item.setName(spInfo.getGysname());
                TbGysinfo gysInfo=Dao.getGysInfo(item);
                item.setId(gysInfo.getId());
                item.setName(gysInfo.getName());
                for (int i=0; i < gysQuanCheng.getItemCount(); i++) {
                    Item gys=(Item) gysQuanCheng.getItemAt(i);
                    if (gys.getName().equals(item.getName())) {
                        item=gys;
                    }
                }
                gysQuanCheng.setSelectedItem(item);
            }
        }
    }
```

19.4.4 进货信息管理

进货信息管理主要包括记录进货单和退货单功能，由于这两个模块的设计十分类似，这里只对进货单功能进行讲解。如图 19-31 所示为该系统的进货单界面。



图 19-31　进货功能

下面通过代码讲解实现该功能的方法：

```
public class JinHuoDan extends JInternalFrame {
  private final JTable table;
  private TbUserlist user=Login.getUser();             // 登录用户信息
  private final JTextField jhsj=new JTextField();      // 进货时间
  private final JTextField jsr=new JTextField();       // 经手人
  private final JComboBox jsfs=new JComboBox();        // 计算方式
  private final JTextField lian=new JTextField();      // 联系人
  private final JComboBox gys=new JComboBox();         // 供应商
  private final JTextField piaoHao=new JTextField();   // 票号
  private final JTextField pzs=new JTextField("0");    // 品种数量
  private final JTextField hpzs=new JTextField("0");   // 货品总数
  private final JTextField hjje=new JTextField("0");   // 合计金额
  private final JTextField ysjl=new JTextField();      // 验收结论
  private final JTextField czy=new JTextField(user.getName());// 操作
员
  private Date jhsjDate;
  private JComboBox sp;
  public JinHuoDan() {
       super();
       setMaximizable(true);
       setIconifiable(true);
       setClosable(true);
       getContentPane().setLayout(new GridBagLayout());
       setTitle("进货单");
       setBounds(50,50,700,400);
       setupComponet(new JLabel("进货票号: "),0,0,1,0,false);
       piaoHao.setFocusable(false);
       setupComponet(piaoHao,1,0,1,140,true);
```

```
setupComponet(new JLabel("供应商："),2,0,1,0,false);
gys.setPreferredSize(new Dimension(160,21));
// 供应商下拉选择框的选择事件
gys.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
          doGysSelectAction();
     }
});
setupComponet(gys,3,0,1,1,true);
setupComponet(new JLabel("联系人："),4,0,1,0,false);
lian.setFocusable(false);
setupComponet(lian,5,0,1,80,true);
setupComponet(new JLabel("结算方式："),0,1,1,0,false);
jsfs.addItem("现金");
jsfs.addItem("支票");
jsfs.setEditable(true);
setupComponet(jsfs,1,1,1,1,true);
setupComponet(new JLabel("进货时间："),2,1,1,0,false);
jhsj.setFocusable(false);
setupComponet(jhsj,3,1,1,1,true);
setupComponet(new JLabel("经手人："),4,1,1,0,false);
setupComponet(jsr,5,1,1,1,true);
sp=new JComboBox();
sp.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
          TbSpinfo info=(TbSpinfo) sp.getSelectedItem();
          // 如果选择有效就更新表格
          if (info !=null && info.getId() != null) {
               updateTable();
          }
     }
});
table=new JTable();
table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
initTable();
// 添加事件完成品种数量、货品总数、合计金额的计算
table.addContainerListener(new computeInfo());
JScrollPane scrollPanel=new JScrollPane(table);
scrollPanel.setPreferredSize(new Dimension(380,200));
setupComponet(scrollPanel,0,2,6,1,true);
setupComponet(new JLabel("品种数量："),0,3,1,0,false);
pzs.setFocusable(false);
setupComponet(pzs,1,3,1,1,true);
setupComponet(new JLabel("货品总数："),2,3,1,0,false);
hpzs.setFocusable(false);
setupComponet(hpzs,3,3,1,1,true);
```

```
                setupComponet(new JLabel("合计金额："),4,3,1,0,false);
                hjje.setFocusable(false);
                setupComponet(hjje,5,3,1,1,true);
                setupComponet(new JLabel("验收结论："),0,4,1,0,false);
                setupComponet(ysjl,1,4,1,1,true);
                setupComponet(new JLabel("操作人员："),2,4,1,0,false);
                czy.setFocusable(false);
                setupComponet(czy,3,4,1,1,true);
                // 单击添加按钮在表格中添加新的一行
                JButton tjButton = new JButton("添加");
                tjButton.addActionListener(new ActionListener() {
                     public void actionPerformed(ActionEvent e) {
                          // 初始化票号
                          initPiaoHao();
                          // 结束表格中没有编写的单元
                          stopTableCellEditing();
                          // 如果表格中还包含空行，就再添加新行
                          for (int i=0; i < table.getRowCount(); i++) {
                               TbSpinfo info=(TbSpinfo) table.getValueAt(i, 0);
                               if (table.getValueAt(i, 0)==null)
                                    return;
                          }
                          DefaultTableModel  model=(DefaultTableModel)  table.
getModel();
                          model.addRow(new Vector());
                          initSpBox();
                     }
                });
                setupComponet(tjButton,4,4,1,1,false);

                // 单击入库按钮保存进货信息
                JButton rkButton=new JButton("入库");
                rkButton.addActionListener(new ActionListener() {
                     public void actionPerformed(ActionEvent e) {
                          // 结束表格中没有编写的单元
                          stopTableCellEditing();
                          // 清除空行
                          clearEmptyRow();
                          String hpzsStr=hpzs.getText(); // 货品总数
                          String pzsStr=pzs.getText(); // 品种数
                          String jeStr=hjje.getText(); // 合计金额
                          String jsfsStr=jsfs.getSelectedItem().toString();
// 结算方式
                          String jsrStr=jsr.getText().trim(); // 经手人
                          String czyStr=czy.getText(); // 操作员
                          String rkDate=jhsjDate.toLocaleString(); // 入库时间
```

```
				String ysjlStr=ysjl.getText().trim(); // 验收结论
				String id=piaoHao.getText(); // 票号
				String gysName=gys.getSelectedItem().toString(); //
供应商名字
				if (jsrStr==null || jsrStr.isEmpty()) {
					JOptionPane.showMessageDialog(JinHuoDan.this,
"请填写经手人");
					return;
				}
				if (ysjlStr==null || ysjlStr.isEmpty()) {
					JOptionPane.showMessageDialog(JinHuoDan.this,
"填写验收结论");
					return;
				}
				if (table.getRowCount()<=0) {
					JOptionPane.showMessageDialog(JinHuoDan.this,
"加入库商品");
					return;
				}
				TbRukuMain ruMain=new TbRukuMain(id, pzsStr, jeStr,
ysjlStr,
						gysName, rkDate, czyStr, jsrStr, jsfsStr);
				Set<TbRukuDetail> set=ruMain.getTabRukuDetails();
				int rows=table.getRowCount();
				for (int i=0; i < rows; i++) {
					TbSpinfo spinfo=(TbSpinfo) table.getValueAt(i, 0);
					String djStr=(String) table.getValueAt(i, 6);
					String slStr=(String) table.getValueAt(i, 7);
					Double dj=Double.valueOf(djStr);
					Integer sl=Integer.valueOf(slStr);
					TbRukuDetail detail=new TbRukuDetail();
					detail.setTabSpinfo(spinfo.getId());
					detail.setTabRukuMain(ruMain.getRkId());
					detail.setDj(dj);
					detail.setSl(sl);
					set.add(detail);
				}
				boolean rs=Dao.insertRukuInfo(ruMain);
				if (rs) {
					JOptionPane.showMessageDialog(JinHuoDan.this,
"入库完成");
					DefaultTableModel dftm=new DefaultTableModel();
					table.setModel(dftm);
					initTable();
					pzs.setText("0");
					hpzs.setText("0");
```

```
                        hjje.setText("0");
                    }
                }
            });
            setupComponet(rkButton,5,4,1,1,false);
            // 添加窗体监听器，完成初始化
            addInternalFrameListener(new initTasks());
        }
        // 初始化表格
        private void initTable() {
            String[] columnNames={"商品名称", "商品编号", "产地", "单位", "规格",
"包装", "单价",
                    "数量", "批号", "批准文号"};
            ((DefaultTableModel) table.getModel())
                    .setColumnIdentifiers(columnNames);
            TableColumn column=table.getColumnModel().getColumn(0);
            final DefaultCellEditor editor=new DefaultCellEditor(sp);
            editor.setClickCountToStart(2);
            column.setCellEditor(editor);
        }
        // 初始化商品下拉选择框
        private void initSpBox() {
            List list=new ArrayList();
            ResultSet set=Dao.query("select * from tb_spinfo where gysName='"
                    + gys.getSelectedItem() + "'");
            sp.removeAllItems();
            sp.addItem(new TbSpinfo());
            for (int i=0; table != null && i < table.getRowCount(); i++) {
                TbSpinfo tmpInfo=(TbSpinfo) table.getValueAt(i, 0);
                if (tmpInfo !=null && tmpInfo.getId() !=null)
                    list.add(tmpInfo.getId());
            }
            try {
                while (set.next()) {
                    TbSpinfo spinfo=new TbSpinfo();
                    spinfo.setId(set.getString("id").trim());
                    // 如果表格中以存在同样商品，商品下拉框中就不再包含该商品
                    if (list.contains(spinfo.getId()))
                        continue;
                    spinfo.setSpname(set.getString("spname").trim());
                    spinfo.setCd(set.getString("cd").trim());
                    spinfo.setJc(set.getString("jc").trim());
                    spinfo.setDw(set.getString("dw").trim());
                    spinfo.setGg(set.getString("gg").trim());
                    spinfo.setBz(set.getString("bz").trim());
                    spinfo.setPh(set.getString("ph").trim());
```

```
                    spinfo.setPzwh(set.getString("pzwh").trim());
                    spinfo.setMemo(set.getString("memo").trim());
                    spinfo.setGysname(set.getString("gysname").trim());
                    sp.addItem(spinfo);
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        // 设置组件位置并添加到容器中
        private void setupComponet(JComponent component, int gridx, int
gridy,
                int gridwidth, int ipadx, boolean fill) {
            final GridBagConstraints gridBagConstrains=new GridBagConst
raints();
            gridBagConstrains.gridx=gridx;
            gridBagConstrains.gridy=gridy;
            if (gridwidth > 1)
                gridBagConstrains.gridwidth=gridwidth;
            if (ipadx > 0)
                gridBagConstrains.ipadx=ipadx;
            gridBagConstrains.insets=new Insets(5,1,3,1);
            if (fill)
                gridBagConstrains.fill=GridBagConstraints.HORIZONTAL;
            getContentPane().add(component, gridBagConstrains);
        }
        // 供应商选择时更新联系人字段
        private void doGysSelectAction() {
            Item item=(Item) gys.getSelectedItem();
            TbGysinfo gysInfo=Dao.getGysInfo(item);
            lian.setText(gysInfo.getLian());
            initSpBox();
        }
        // 在事件中计算品种数量、货品总数、合计金额
        private final class computeInfo implements ContainerListener {
            public void componentRemoved(ContainerEvent e) {
                // 清除空行
                clearEmptyRow();
                // 计算代码
                int rows=table.getRowCount();
                int count=0;
                double money=0.0;
                // 计算品种数量
                TbSpinfo column=null;
                if (rows > 0)
                    column=(TbSpinfo) table.getValueAt(rows - 1,0);
```

```
                    if (rows > 0 && (column==null || column.getId().isEmpty()))
                         rows--;
                    // 计算货品总数和金额
                    for (int i=0; i < rows; i++) {
                         String column7=(String) table.getValueAt(i,7);
                         String column6=(String) table.getValueAt(i,6);
                         int  c7=(column7==null  ||  column7.isEmpty())  ?  0  :
Integer
                                   .parseInt(column7);
                         float  c6=(column6==null  ||  column6.isEmpty()) ?  0  :
Float
                                   .parseFloat(column6);
                         count += c7;
                         money += c6 * c7;
                    }
                    pzs.setText(rows + "");
                    hpzs.setText(count + "");
                    hjje.setText(money + "");
                              }
               public void componentAdded(ContainerEvent e) {
               }
          }
          // 窗体的初始化任务
          private final class initTasks extends InternalFrameAdapter {
               public void internalFrameActivated(InternalFrameEvent e) {
                    super.internalFrameActivated(e);
                    initTimeField();
                    initGysField();
                    initPiaoHao();
                    initSpBox();
               }
               private void initGysField() {// 初始化供应商字段
                    List gysInfos=Dao.getGysInfos();
                    for (Iterator iter=gysInfos.iterator(); iter.hasNext();) {
                         List list=(List) iter.next();
                         Item item=new Item();
                         item.setId(list.get(0).toString().trim());
                         item.setName(list.get(1).toString().trim());
                         gys.addItem(item);
                    }
                    doGysSelectAction();
               }
               private void initTimeField() {// 启动进货时间线程
                    new Thread(new Runnable()
        {
                         public void run()
```

```
 {
                     try {
                          while (true)
        {
                               jhsjDate=new Date();
                               jhsj.setText(jhsjDate.toLocaleString());
                               Thread.sleep(100);
                          }
                     } catch (InterruptedException e)
        {
                          e.printStackTrace();
                     }
                }
           }).start();
      }
   }
   // 初始化票号文本框的方法
   private void initPiaoHao()
        {
      java.sql.Date date=new java.sql.Date(jhsjDate.getTime());
      String maxId=Dao.getRuKuMainMaxId(date);
      piaoHao.setText(maxId);
   }
   // 根据商品下拉框的选择，更新表格当前行的内容
   private synchronized void updateTable()
        {
      TbSpinfo spinfo=(TbSpinfo) sp.getSelectedItem();
      int row=table.getSelectedRow();
      if (row >=0 && spinfo !=null)
        {
           table.setValueAt(spinfo.getId(),row,1);
           table.setValueAt(spinfo.getCd(),row,2);
           table.setValueAt(spinfo.getDw(),row,3);
           table.setValueAt(spinfo.getGg(),row,4);
           table.setValueAt(spinfo.getBz(),row,5);
           table.setValueAt("0",row,6);
           table.setValueAt("0",row,7);
           table.setValueAt(spinfo.getPh(),row,8);
           table.setValueAt(spinfo.getPzwh(),row,9);
           table.editCellAt(row,6);
      }
   }
   // 清除空行
   private synchronized void clearEmptyRow()
 {
      DefaultTableModel dftm=(DefaultTableModel) table.getModel();
```

```
            for (int i = 0; i < table.getRowCount(); i++)
    {
                TbSpinfo info2=(TbSpinfo) table.getValueAt(i,0);
                if (info2==null || info2.getId()==null
                        || info2.getId().isEmpty())
    {
                    dftm.removeRow(i);
                }
            }
      }
      // 停止表格单元的编辑
      private void stopTableCellEditing()
    {
            TableCellEditor cellEditor=table.getCellEditor();
            if (cellEditor !=null)
                cellEditor.stopCellEditing();
      }
    }
```

19.4.5 销售信息管理

销售信息管理是系统中的一个重要模块，这个模块主要处理销售单和销售退货信息。这两个功能实现的方式基本相同，这里只讲解销售单功能。如图 19-32 所示为本系统的销售单管理界面。



图 19-32 销售单

下面通过代码进行讲解：

```
    public class XiaoShouDan extends JInternalFrame {
      private final JTable table;
```

```
        private TbUserlist user=Login.getUser(); // 登录用户信息
        private final JTextField jhsj=new JTextField(); // 进货时间
        private final JTextField jsr=new JTextField(); // 经手人
        private final JComboBox jsfs=new JComboBox(); // 计算方式
        private final JTextField lian=new JTextField(); // 联系人
        private final JComboBox kehu=new JComboBox(); // 客户
        private final JTextField piaoHao=new JTextField(); // 票号
        private final JTextField pzs=new JTextField("0"); // 品种数量
        private final JTextField hpzs=new JTextField("0"); // 货品总数
        private final JTextField hjje=new JTextField("0"); // 合计金额
        private final JTextField ysjl=new JTextField(); // 验收结论
        private final JTextField czy=new JTextField(user.getName());// 操作
员
        private Date jhsjDate;
        private JComboBox sp;
        public XiaoShouDan() {
            super();
            setMaximizable(true);
            setIconifiable(true);
            setClosable(true);
            getContentPane().setLayout(new GridBagLayout());
            setTitle("销售单");
            setBounds(50,50,700,400);
            setupComponet(new JLabel("销售票号："),0,0,1,0,false);
            piaoHao.setFocusable(false);
            setupComponet(piaoHao,1,0,1,140,true);
            setupComponet(new JLabel("客户："),2,0,1,0,false);
            kehu.setPreferredSize(new Dimension(160, 21));
            // 供应商下拉选择框的选择事件
            kehu.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    doKhSelectAction();
                }
            });
            setupComponet(kehu,3,0,1,1,true);
            setupComponet(new JLabel("联系人："),4,0,1,0,false);
            lian.setFocusable(false);
            lian.setPreferredSize(new Dimension(80,21));
            setupComponet(lian,5,0,1,0,true);
            setupComponet(new JLabel("结算方式："),0,1,1,0,false);
            jsfs.addItem("现金");
            jsfs.addItem("支票");
            jsfs.setEditable(true);
            setupComponet(jsfs,1,1,1,1,true);
            setupComponet(new JLabel("销售时间："),2,1,1,0,false);
            jhsj.setFocusable(false);
```

```
                setupComponet(jhsj,3,1,1,1,true);
                setupComponet(new JLabel("经手人："),4,1,1,0,false);
                setupComponet(jsr,5,1,1,1,true);
                sp=new JComboBox();
                sp.addActionListener(new ActionListener() {
                     public void actionPerformed(ActionEvent e) {
                          TbSpinfo info=(TbSpinfo) sp.getSelectedItem();
                          // 如果选择有效就更新表格
                          if (info != null && info.getId() != null) {
                               updateTable();
                          }
                     }
                });
                table=new JTable();
                table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
                initTable();
                // 添加事件完成品种数量、货品总数、合计金额的计算
                table.addContainerListener(new computeInfo());
                JScrollPane scrollPanel=new JScrollPane(table);
                scrollPanel.setPreferredSize(new Dimension(380, 200));
                setupComponet(scrollPanel,0,2,6,1,true);
                setupComponet(new JLabel("品种数量："),0,3,1,0,false);
                pzs.setFocusable(false);
                setupComponet(pzs,1,3,1,1,true);
                setupComponet(new JLabel("货品总数："),2,3,1,0,false);
                hpzs.setFocusable(false);
                setupComponet(hpzs,3,3,1,1,true);
                setupComponet(new JLabel("合计金额："),4,3,1,0,false);
                hjje.setFocusable(false);
                setupComponet(hjje,5,3,1,1,true);
                setupComponet(new JLabel("验收结论："),0,4,1,0,false);
                setupComponet(ysjl,1,4,1,1,true);
                setupComponet(new JLabel("操作人员："),2,4,1,0,false);
                czy.setFocusable(false);
                setupComponet(czy,3,4,1,1,true);
                // 单击添加按钮在表格中添加新的一行
                JButton tjButton=new JButton("添加");
                tjButton.addActionListener(new ActionListener() {
                     public void actionPerformed(ActionEvent e) {
                          // 初始化票号
                          initPiaoHao();
                          // 结束表格中没有编写的单元
                          stopTableCellEditing();
                          // 如果表格中还包含空行，就再添加新行
                          for (int i=0; i < table.getRowCount(); i++) {
                               TbSpinfo info=(TbSpinfo) table.getValueAt(i, 0);
```

```
                        if (table.getValueAt(i, 0)==null)
                            return;
                    }
                    DefaultTableModel model=(DefaultTableModel) table. get
Model();
                    model.addRow(new Vector());
                }
            });
            setupComponet(tjButton,4,4,1,1,false);
            // 单击销售按钮保存进货信息
            JButton sellButton=new JButton("销售");
            sellButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    stopTableCellEditing();    // 结束表格中没有编写的单元
                    clearEmptyRow();           // 清除空行
                    String hpzsStr=hpzs.getText();  // 货品总数
                    String pzsStr=pzs.getText();           // 品种数
                    String jeStr=hjje.getText();           // 合计金额
                    String   jsfsStr=jsfs.getSelectedItem().toString();
// 结算方式
                    String jsrStr=jsr.getText().trim(); // 经手人
                    String czyStr=czy.getText();           // 操作员
                    String rkDate=jhsjDate.toLocaleString();
// 销售时间
                    String ysjlStr=ysjl.getText().trim(); // 验收结论
                    String id=piaoHao.getText();           // 票号
                    String kehuName=kehu.getSelectedItem().toString
();// 供应商名字
                    if (jsrStr==null || jsrStr.isEmpty()) {
                        JOptionPane.showMessageDialog(XiaoShouDan.this,
"请填写经手人");
                        return;
                    }
                    if (ysjlStr==null || ysjlStr.isEmpty()) {
                        JOptionPane.showMessageDialog(XiaoShouDan.this,
"填写验收结论");
                        return;
                    }
                    if (table.getRowCount() <=0) {
                        JOptionPane.showMessageDialog(XiaoShouDan.this,
"填加销售商品");
                        return;
                    }
                    TbSellMain sellMain=new TbSellMain(id, pzsStr, jeStr,
                            ysjlStr, kehuName, rkDate, czyStr, jsrStr,
jsfsStr);
```

```
                    Set<TbSellDetail> set=sellMain.getTbSellDetails();
                    int rows=table.getRowCount();
                    for (int i=0; i < rows; i++) {
                         TbSpinfo spinfo=(TbSpinfo) table.getValueAt(i,0);
                         String djStr=(String) table.getValueAt(i,6);
                         String slStr=(String) table.getValueAt(i,7);
                         Double dj=Double.valueOf(djStr);
                         Integer sl=Integer.valueOf(slStr);
                         TbSellDetail detail=new TbSellDetail();
                         detail.setSpid(spinfo.getId());
                         detail.setTbSellMain(sellMain.getSellId());
                         detail.setDj(dj);
                         detail.setSl(sl);
                         set.add(detail);
                    }
                    boolean rs=Dao.insertSellInfo(sellMain);
                    if (rs) {
                         JOptionPane.showMessageDialog(XiaoShouDan.this,
"销售完成");
                         DefaultTableModel dftm=new DefaultTableModel();
                         table.setModel(dftm);
                         initTable();
                         pzs.setText("0");
                         hpzs.setText("0");
                         hjje.setText("0");
                    }
               }
          });
          setupComponet(sellButton,5,4,1,1,false);
          // 添加窗体监听器，完成初始化
          addInternalFrameListener(new initTasks());
     }
     // 初始化表格
     private void initTable() {
String[] columnNames={"商品名称", "商品编号", "供应商", "产地", "单位", "规格", "单
价",
                    "数量", "包装", "批号", "批准文号"};
          ((DefaultTableModel) table.getModel())
                    .setColumnIdentifiers(columnNames);
          TableColumn column=table.getColumnModel().getColumn(0);
          final DefaultCellEditor editor=new DefaultCellEditor(sp);
          editor.setClickCountToStart(2);
          column.setCellEditor(editor);
     }
     // 初始化商品下拉选择框
     private void initSpBox() {
```

```
        List list=new ArrayList();
        ResultSet set=Dao.query("select * from tb_spinfo"
                 + " where id in (select id from tb_kucun where kcsl>0)");
        sp.removeAllItems();
        sp.addItem(new TbSpinfo());
        for (int i=0; table !=null && i < table.getRowCount(); i++) {
             TbSpinfo tmpInfo=(TbSpinfo) table.getValueAt(i,0);
             if (tmpInfo !=null && tmpInfo.getId() !=null)
                  list.add(tmpInfo.getId());
        }
        try {
             while (set.next()) {
                  TbSpinfo spinfo=new TbSpinfo();
                  spinfo.setId(set.getString("id").trim());
                  // 如果表格中以存在同样商品，商品下拉框中就不再包含该商品
                  if (list.contains(spinfo.getId()))
                       continue;
                  spinfo.setSpname(set.getString("spname").trim());
                  spinfo.setCd(set.getString("cd").trim());
                  spinfo.setJc(set.getString("jc").trim());
                  spinfo.setDw(set.getString("dw").trim());
                  spinfo.setGg(set.getString("gg").trim());
                  spinfo.setBz(set.getString("bz").trim());
                  spinfo.setPh(set.getString("ph").trim());
                  spinfo.setPzwh(set.getString("pzwh").trim());
                  spinfo.setMemo(set.getString("memo").trim());
                  spinfo.setGysname(set.getString("gysname").trim());
                  sp.addItem(spinfo);
             }
        } catch (SQLException e) {
             e.printStackTrace();
        }
   }
   // 设置组件位置并添加到容器中
   private  void  setupComponet(JComponent  component,  int  gridx,  int
gridy,
             int gridwidth, int ipadx, boolean fill) {
        final GridBagConstraints gridBagConstrains=new GridBagConstraints();
        gridBagConstrains.gridx=gridx;
        gridBagConstrains.gridy=gridy;
        if (gridwidth>1)
             gridBagConstrains.gridwidth=gridwidth;
        if (ipadx>0)
             gridBagConstrains.ipadx=ipadx;
        gridBagConstrains.insets=new Insets(5,1,3,1);
        if (fill)
             gridBagConstrains.fill=GridBagConstraints.HORIZONTAL;
        getContentPane().add(component, gridBagConstrains);
   }
   // 供应商选择时更新联系人字段
```

```
        private void doKhSelectAction() {
            Item item=(Item) kehu.getSelectedItem();
            TbKhinfo khInfo=Dao.getKhInfo(item);
            lian.setText(khInfo.getLian());
        }
        // 在事件中计算品种数量、货品总数、合计金额
        private final class computeInfo implements ContainerListener {
            public void componentRemoved(ContainerEvent e) {
                // 清除空行
                clearEmptyRow();
                // 计算代码
                int rows=table.getRowCount();
                int count=0;
                double money=0.0;
                // 计算品种数量
                TbSpinfo column=null;
                if (rows>0)
                    column=(TbSpinfo) table.getValueAt(rows-1,0);
                if (rows>0 && (column==null || column.getId().isEmpty()))
                    rows--;
                // 计算货品总数和金额
                for (int i=0; i<rows; i++) {
                    String column7=(String) table.getValueAt(i,7);
                    String column6=(String) table.getValueAt(i,6);
                    int  c7=(column7==null  ||  column7.isEmpty())  ?  0  :
Integer
                                .valueOf(column7);
                    Double  c6=(column6==null  ||  column6.isEmpty())  ?  0  :
Double
                                .valueOf(column6);
                    count+=c7;
                    money+=c6 * c7;
                }
                pzs.setText(rows+"");
                hpzs.setText(count+"");
                hjje.setText(money+"");
                            }
            public void componentAdded(ContainerEvent e) {
            }
        }
        // 窗体的初始化任务
        private final class initTasks extends InternalFrameAdapter {
            public void internalFrameActivated(InternalFrameEvent e) {
                super.internalFrameActivated(e);
                initTimeField();
                initKehuField();
                initPiaoHao();
                initSpBox();
            }
```

```
            private void initKehuField() {// 初始化客户字段
                List gysInfos=Dao.getKhInfos();
                for (Iterator iter=gysInfos.iterator(); iter.hasNext();) {
                    List list=(List) iter.next();
                    Item item=new Item();
                    item.setId(list.get(0).toString().trim());
                    item.setName(list.get(1).toString().trim());
                    kehu.addItem(item);
                }
                doKhSelectAction();
            }
            private void initTimeField() {// 启动进货时间线程
                new Thread(new Runnable() {
                    public void run() {
                        try {
                            while (true) {
                                jhsjDate=new Date();
                                jhsj.setText(jhsjDate.toLocaleString
());

                                Thread.sleep(100);
                            }
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }).start();
            }
        }
        private void initPiaoHao() {
            java.sql.Date date=new java.sql.Date(jhsjDate.getTime());
            String maxId=Dao.getSellMainMaxId(date);
            piaoHao.setText(maxId);
        }
        // 根据商品下拉框的选择，更新表格当前行的内容
        private synchronized void updateTable() {
            TbSpinfo spinfo=(TbSpinfo) sp.getSelectedItem();
            Item item=new Item();
            item.setId(spinfo.getId());
            TbKucun kucun=Dao.getKucun(item);
            int row=table.getSelectedRow();
            if (row >=0 && spinfo !=null) {
                table.setValueAt(spinfo.getId(),row,1);
                table.setValueAt(spinfo.getGysname(),row,2);
                table.setValueAt(spinfo.getCd(),row,3);
                table.setValueAt(spinfo.getDw(),row,4);
                table.setValueAt(spinfo.getGg(),row,5);
                table.setValueAt(kucun.getDj()+"",row,6);
                table.setValueAt(kucun.getKcsl()+"",row,7);
                table.setValueAt(spinfo.getBz(),row,8);
```

```
                table.setValueAt(spinfo.getPh(),row,9);
                table.setValueAt(spinfo.getPzwh(),row,10);
                table.editCellAt(row, 7);
            }
        }
        // 清除空行
        private synchronized void clearEmptyRow() {
            DefaultTableModel dftm=(DefaultTableModel) table.getModel();
            for (int i=0; i <table.getRowCount(); i++) {
                TbSpinfo info2=(TbSpinfo) table.zetValueAt(i,0);
                if (info2==null || info2.getId()==null
                        || info2.getId().isEmpty()) {
                    dftm.removeRow(i);
                }
            }
        }
        // 停止表格单元的编辑
        private void stopTableCellEditing() {
            TableCellEditor cellEditor=table.getCellEditor();
            if (cellEditor !=null)
                cellEditor.stopCellEditing();
        }
    }
```

19.4.6 库存管理

这个系统的库存管理功能主要用来盘点库存和调整库存商品价格的，下面对它进行讲解。

（1）库存盘点功能

库存盘点功能可以更好地帮助管理人员整理库存商品的信息，其制作完成后的效果如图 19-33 所示。



图 19-33 库存盘点

下面通过代码进行讲解：

```
public class KuCunPanDian extends JInternalFrame {
    private final JTable table;
    private TbUserlist user=Login.getUser(); // 登录用户信息
    private final JTextField pdsj=new JTextField(); // 进货时间
    private final JTextField pzs=new JTextField("0"); // 品种数量
    private final JTextField hpzs=new JTextField("0"); // 货品总数
    private final JTextField kcje=new JTextField("0"); // 库存金额
    private Date pdDate=new Date();
    private JTextField pdy=new JTextField(user.getUsername());// 盘点员
    public KuCunPanDian() {
        super();
        setMaximizable(true);
        setIconifiable(true);
        setClosable(true);
        getContentPane().setLayout(new GridBagLayout());
        setTitle("库存盘点");
        setBounds(50,50,750,400);
        setupComponet(new JLabel("盘 点 员："),0,0,1,0,false);
        pdy.setFocusable(false);
        pdy.setPreferredSize(new Dimension(120,21));
        setupComponet(pdy,1,0,1,0,true);
        setupComponet(new JLabel("盘点时间："),2,0,1,0,false);
        pdsj.setFocusable(false);
        pdsj.setText(pdDate.toLocaleString());
        pdsj.setPreferredSize(new Dimension(180,21));
        setupComponet(pdsj,3,0,1,1,true);
        setupComponet(new JLabel("品 种 数："),4,0,1,0,false);
        pzs.setFocusable(false);
        pzs.setPreferredSize(new Dimension(80,21));
        setupComponet(pzs,5,0,1,20,true);
        table=new JTable();
        table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
        initTable();
        JScrollPane scrollPanel=new JScrollPane(table);
        scrollPanel.setPreferredSize(new Dimension(700,300));
        setupComponet(scrollPanel,0,2,6,1,true);
    }
    // 初始化表格
    private void initTable() {
        String[] columnNames={"商品名称","商品编号","供应商","产地","单位
","规格","单价",
                "数量","包装","盘点数量","损益数量"};
        DefaultTableModel tableModel=(DefaultTableModel) table.getModel ();
        tableModel.setColumnIdentifiers(columnNames);
        // 设置盘点字段只接收数字输入
        final JTextField pdField=new JTextField(0);
        pdField.setEditable(false);
```

```
                pdField.addKeyListener(new KeyAdapter() {
                     public void keyTyped(KeyEvent e) {
                          if (("0123456789"+(char) 8).indexOf(e.getKeyChar()+"") < 0)
{
                               e.consume();
                          }
                          pdField.setEditable(true);
                     }
                     public void keyReleased(KeyEvent e) {
                          String pdStr=pdField.getText();
                          String kcStr="0";
                          int row=table.getSelectedRow();
                          if (row>=0) {
                               kcStr=(String) table.getValueAt(row,7);
                          }
                          try {
                               int pdNum=Integer.parseInt(pdStr);
                               int kcNum=Integer.parseInt(kcStr);
                               if (row >=0) {
                                    table.setValueAt(kcNum-pdNum,row,10);
                               }
                               if (e.getKeyChar()!=8)
                                    pdField.setEditable(false);
                          } catch (NumberFormatException e1) {
                               pdField.setText("0");
                          }
                     }
                });
                JTextField readOnlyField=new JTextField(0);
                readOnlyField.setEditable(false);
                DefaultCellEditor pdEditor=new DefaultCellEditor(pdField);
                DefaultCellEditor  readOnlyEditor=new  DefaultCellEditor(readOnly
Field);
                // 设置表格单元为只读格式
                for (int i=0; i<columnNames.length; i++) {
                    TableColumn column=table.getColumnModel().getColumn(i);
                    column.setCellEditor(readOnlyEditor);
                }
                TableColumn pdColumn=table.getColumnModel().getColumn(9);
                TableColumn syColumn=table.getColumnModel().getColumn(10);
                pdColumn.setCellEditor(pdEditor);
                syColumn.setCellEditor(readOnlyEditor);
                // 初始化表格内容
                List kcInfos=Dao.getKucunInfos();
                for (int i=0; i < kcInfos.size(); i++) {
                     List info=(List) kcInfos.get(i);
                     Item item=new Item();
```

```
                    item.setId((String) info.get(0));
                    item.setName((String) info.get(1));
                    TbSpinfo spinfo=Dao.getSpInfo(item);
                    Object[] row=new Object[columnNames.length];
                    if (spinfo.getId() !=null && !spinfo.getId().isEmpty()) {
                         row[0]=spinfo.getSpname();
                         row[1]=spinfo.getId();
                         row[2]=spinfo.getGysname();
                         row[3]=spinfo.getCd();
                         row[4]=spinfo.getDw();
                         row[5]=spinfo.getGg();
                         row[6]=info.get(2).toString();
                         row[7]=info.get(3).toString();
                         row[8]=spinfo.getBz();
                         row[9]=0;
                         row[10]=0;
                         tableModel.addRow(row);
                         String pzsStr=pzs.getText();
                         int pzsInt=Integer.parseInt(pzsStr);
                         pzsInt++;
                         pzs.setText(pzsInt+"");
                    }
               }
          }
          // 设置组件位置并添加到容器中
          private void setupComponet(JComponent component, int gridx, int
gridy,
                    int gridwidth, int ipadx, boolean fill) {
               final GridBagConstraints gridBagConstrains=new GridBagConstraints
();
               gridBagConstrains.gridx=gridx;
               gridBagConstrains.gridy=gridy;
               if (gridwidth>1)
                    gridBagConstrains.gridwidth=gridwidth;
               if (ipadx>0)
                    gridBagConstrains.ipadx=ipadx;
               gridBagConstrains.insets=new Insets(5,1,3,5);
               if (fill)
                    gridBagConstrains.fill=GridBagConstraints.HORIZONTAL;
               getContentPane().add(component, gridBagConstrains);
          }
     }
```

（2）库存价格调整

使用库存价格调整功能可以对库存的商品价格进行修改，制作完成后的效果如图 19-34 所示。

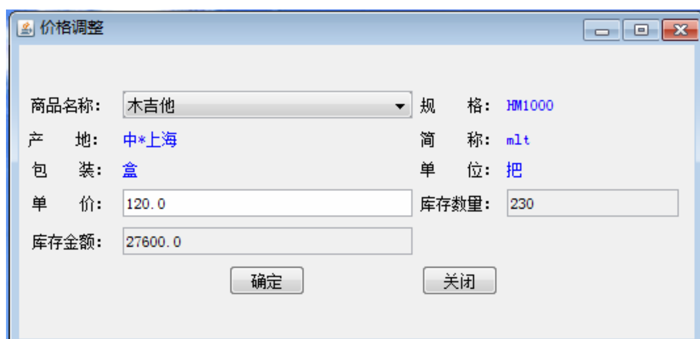


图 19-34　价格调整

下面通过代码进行讲解：

```
        public class JiaGeTiaoZheng extends JInternalFrame {
           private TbKucun kcInfo;
           private JLabel guiGe;
           private JTextField kuCunJinE;
           private JTextField kuCunShuLiang;
           private JTextField danJia;
           private JComboBox shangPinMingCheng;
           private void updateJinE() {
                Double dj=Double.valueOf(danJia.getText());
                Integer sl=Integer.valueOf(kuCunShuLiang.getText());
                kuCunJinE.setText((dj*sl)+"");
           }
           public JiaGeTiaoZheng() {
                super();
                addInternalFrameListener(new InternalFrameAdapter() {
                     public  void  internalFrameActivated(final  InternalFrameEvent
e) {
                          DefaultComboBoxModel mingChengModel=(DefaultComboBox
Model) shangPinMingCheng
                                    .getModel();
                          mingChengModel.removeAllElements();
                          List list=Dao.getKucunInfos();
                          Iterator iterator=list.iterator();
                          while (iterator.hasNext()) {
                               List element=(List) iterator.next();
                               Item item=new Item();
                               item.setId((String) element.get(0));
                               item.setName((String) element.get(1));
                               mingChengModel.addElement(item);
                          }
                     }
                });
                setMaximizable(true);
```

```
setIconifiable(true);
setClosable(true);
getContentPane().setLayout(new GridBagLayout());
setTitle("价格调整");
setBounds(100,100,531,253);
setupComponet(new JLabel("商品名称："),0,0,1,1,false);
shangPinMingCheng=new JComboBox();
shangPinMingCheng.setPreferredSize(new Dimension(220,21));
setupComponet(shangPinMingCheng,1,0,1,1,true);
setupComponet(new JLabel("规    格："),2,0,1,0,false);
guiGe=new JLabel();
guiGe.setForeground(Color.BLUE);
guiGe.setPreferredSize(new Dimension(130,21));
setupComponet(guiGe,3,0,1,1,true);
setupComponet(new JLabel("产    地： "),0,1,1,0,false);
final JLabel chanDi=new JLabel();
chanDi.setForeground(Color.BLUE);
setupComponet(chanDi,1,1,1,1,true);
setupComponet(new JLabel("简    称："),2,1,1,0,false);
final JLabel jianCheng=new JLabel();
jianCheng.setForeground(Color.BLUE);
setupComponet(jianCheng,3,1,1,1,true);
setupComponet(new JLabel("包    装："),0,2,1,0,false);
final JLabel baoZhuang=new JLabel();
baoZhuang.setForeground(Color.BLUE);
setupComponet(baoZhuang,1,2,1,1,true);
setupComponet(new JLabel("单    位："),2,2,1,0,false);
final JLabel danWei=new JLabel();
danWei.setForeground(Color.BLUE);
setupComponet(danWei,3,2,1,1,true);
         setupComponet(new JLabel("单    价："),0,3,1,0,false);
danJia=new JTextField();
danJia.addKeyListener(new KeyAdapter() {
     public void keyReleased(final KeyEvent e) {
          updateJinE();
     }
});
setupComponet(danJia,1,3,1,1,true);
setupComponet(new JLabel("库存数量："),2,3,1,0,false);
kuCunShuLiang=new JTextField();
kuCunShuLiang.setEditable(false);
setupComponet(kuCunShuLiang,3,3,1,1,true);
setupComponet(new JLabel("库存金额："),0,4,1,0,false);
kuCunJinE=new JTextField();
kuCunJinE.setEditable(false);
```

```
                setupComponet(kuCunJinE,1,4,1,1,true);

                final JButton okButton=new JButton();
                okButton.addActionListener(new ActionListener() {
                     public void actionPerformed(final ActionEvent e) {
                          kcInfo.setDj(Double.valueOf(danJia.getText()));

     kcInfo.setKcsl(Integer.valueOf(kuCunShuLiang.getText()));
                          int rs=Dao.updateKucunDj(kcInfo);
                          if (rs>0)
                               JOptionPane.showMessageDialog(getContentPane(),
"价格调整完毕。",
                                          kcInfo.getSpname()+"价格调整",
                                          JOptionPane.QUESTION_MESSAGE);
                     }
                });
                okButton.setText("确定");
                setupComponet(okButton,1,5,1,1,false);

                final JButton closeButton=new JButton();
                closeButton.addActionListener(new ActionListener() {
                     public void actionPerformed(final ActionEvent e) {
                          JiaGeTiaoZheng.this.doDefaultCloseAction();
                     }
                });
                closeButton.setText("关闭");
                setupComponet(closeButton,2,5,1,1,false);

                shangPinMingCheng.addItemListener(new ItemListener()
        {
                     public void itemStateChanged(final ItemEvent e)
        {
                          Object selectedItem=shangPinMingCheng.getSelected Item();
                          if (selectedItem==null)
                               return;
                          Item item=(Item) selectedItem;
                          kcInfo=Dao.getKucun(item);
                          if(kcInfo.getId()==null)
                               return;
                          int dj, sl;
                          dj=kcInfo.getDj().intValue();
                          sl=kcInfo.getKcsl().intValue();
                          chanDi.setText(kcInfo.getCd());
                          jianCheng.setText(kcInfo.getJc());
                          baoZhuang.setText(kcInfo.getBz());
                          danWei.setText(kcInfo.getDw());
```

```
                        danJia.setText(kcInfo.getDj()+"");
                        kuCunShuLiang.setText(kcInfo.getKcsl()+"");
                        kuCunJinE.setText(dj * sl+"");
                        guiGe.setText(kcInfo.getGg());
                    }
                });
            }
            // 设置组件位置并添加到容器中
            private void setupComponet(JComponent component, int gridx, int gridy,
                    int gridwidth, int ipadx, boolean fill)
        {
                final GridBagConstraints gridBagConstrains=new GridBagConstraints();
                gridBagConstrains.gridx=gridx;
                gridBagConstrains.gridy=gridy;
                if (gridwidth>1)
                    gridBagConstrains.gridwidth=gridwidth;
                if (ipadx>0)
                    gridBagConstrains.ipadx=ipadx;
                gridBagConstrains.insets=new Insets(5,1,3,5);
                if (fill)
                    gridBagConstrains.fill=GridBagConstraints.HORIZONTAL;
                getContentPane().add(component, gridBagConstrains);
            }
        }
```

19.4.7 查询与统计

查询与统计是书城系统中最常用的功能，其中包括了客户信息查询、供应商信息查询、商品信息查询、销售信息查询、销售退货查询、入库查询、入库退货查询以及销售排行功能。这里主要讲解销售排行功能，销售排行功能是用来查询销售排行信息的，并可以根据用户的选择将信息按照字段进行升序和降序排列。制作完成后的效果如图 19-35 所示。



图 19-35 销售排行

下面通过代码进行讲解：

```
        public class XiaoShouPaiHang extends JInternalFrame {
            private JButton okButton;
            private JComboBox month;
```

```
        private JComboBox year;
        private JTable table;
        private JComboBox operation;
        private JComboBox condition;
        private TbUserlist user;
        private DefaultTableModel dftm;
        private Calendar date=Calendar.getInstance();
        public XiaoShouPaiHang() {
            setIconifiable(true);
            setClosable(true);
            setTitle("销售排行");
            getContentPane().setLayout(new GridBagLayout());
            setBounds(100,100,650,375);

            final JLabel label_1=new JLabel();
            label_1.setText("对");
            final GridBagConstraints gridBagConstraints_8 = new GridBagCon
straints();
            gridBagConstraints_8.anchor=GridBagConstraints.EAST;
            gridBagConstraints_8.gridy=0;
            gridBagConstraints_8.gridx=0;
            getContentPane().add(label_1, gridBagConstraints_8);

            year=new JComboBox();
            for (int i=1981, j=0; i<=date.get(Calendar.YEAR) + 1; i++, j++) {
                year.addItem(i);
                if (i==date.get(Calendar.YEAR))
                    year.setSelectedIndex(j);
            }
            year.setPreferredSize(new Dimension(100,21));
            setupComponet(year,1,0,1,90,true);

            setupComponet(new JLabel("到"),2,0,1,1,false);

            month=new JComboBox();
            for (int i=1; i<=12; i++) {
                month.addItem(String.format("%02d",i));
                if (date.get(Calendar.MONTH)==i)
                    month.setSelectedIndex(i-1);
            }
            month.setPreferredSize(new Dimension(100,21));
            setupComponet(month,3,0,1,30,true);

            setupComponet(new JLabel(" 月份的销售信息，按"),4,0,1,1,false);
            condition=new JComboBox();
            condition.setModel(new DefaultComboBoxModel(new String[]{"金额
```

```
", "数量"}));
                setupComponet(condition,5,0,1,30,true);

                setupComponet(new JLabel(" 进行"),6,0,1,1,false);

                operation=new JComboBox();
                operation.setModel(new DefaultComboBoxModel(
                            new String[]{"升序排列","降序排列"}));
                setupComponet(operation,7,0,1,30,true);

                okButton=new JButton();
                okButton.addActionListener(new OkAction());
                setupComponet(okButton,8,0,1,1,false);
                okButton.setText("确定");

                final JScrollPane scrollPane=new JScrollPane();
                final  GridBagConstraints  gridBagConstraints_6=new  GridBagCon
straints();
                gridBagConstraints_6.weighty=1.0;
                gridBagConstraints_6.anchor=GridBagConstraints.NORTH;
                gridBagConstraints_6.insets=new Insets(0,10,5,10);
                gridBagConstraints_6.fill=GridBagConstraints.BOTH;
                gridBagConstraints_6.gridwidth=9;
                gridBagConstraints_6.gridy=1;
                gridBagConstraints_6.gridx=0;
                getContentPane().add(scrollPane, gridBagConstraints_6);

                table=new JTable();
                table.setEnabled(false);
                table.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
                dftm=(DefaultTableModel) table.getModel();
               String[] tableHeads=new String[]{"商品编号", "商品名称", "销售金额",
"销售数量",
                            "简称", "产地", "单位", "规格", "包装", "批号", "批准文号",
"简介","供应商"};
                dftm.setColumnIdentifiers(tableHeads);
                scrollPane.setViewportView(table);
          }

          private void updateTable(Iterator iterator) {
                int rowCount=dftm.getRowCount();
                for (int i=0; i < rowCount; i++) {
                      dftm.removeRow(0);
                }
                while (iterator.hasNext()) {
```

```
                Vector vector=new Vector();
                List view=(List) iterator.next();
                Vector row=new Vector(view);
                int rowSize=row.size();
                for(int i=rowSize-2;i<rowSize;i++){
                    Object colValue=row.get(i);
                    row.remove(i);
                    row.insertElementAt(colValue, 2);
                }
                vector.addAll(row);
                dftm.addRow(vector);
            }
        }
        // 设置组件位置并添加到容器中
        private void setupComponet(JComponent component, int gridx, int
gridy,
                int gridwidth, int ipadx, boolean fill) {
            final GridBagConstraints gridBagConstrains=new GridBagCon
straints();
            gridBagConstrains.gridx=gridx;
            gridBagConstrains.gridy=gridy;
            if (gridwidth > 1)
                gridBagConstrains.gridwidth=gridwidth;
            if (ipadx > 0)
                gridBagConstrains.ipadx=ipadx;
            gridBagConstrains.insets=new Insets(5,1,3,1);
            if (fill)
                gridBagConstrains.fill=GridBagConstraints.HORIZONTAL;
            getContentPane().add(component, gridBagConstrains);
        }
        private final class OkAction implements ActionListener {
            public void actionPerformed(final ActionEvent e) {
                List list=null;
                String strMonth=(String) month.getSelectedItem();
                String date=year.getSelectedItem()+strMonth;
                String con=condition.getSelectedIndex()==0 ? "sumje " :
"sl ";
                int oper=operation.getSelectedIndex();
                String sql1="select spid,sum(sl)as sl,sum(sl*dj) as sumje
from"
                        +" v_sellView where substring(convert(varchar
(30)"
                        + ",xsdate,112),0,7)='" + date+"' group by
spid";
                String opstr=oper==0 ? " asc" : " desc";
                String queryStr="select * from tb_spinfo s inner join
```

```
("+ sql1
                              +") as sp on s.id=sp.spid order by " + con +
opstr;
                    list=Dao.findForList(queryStr);
                    Iterator iterator=list.iterator();
                    updateTable(iterator);
               }
          }
        }
```

19.5 数据库模块的编程

若要正确运行上面的代码，必须要编写一个类，它需要加载数据库驱动，然后连接数据库，否则无法运行。除此之外，它还将处理一些数据库，如添加商品、编写修改商品等等信息，下面通过代码进行讲解：

```
        public class Dao {
          protected static String dbClassName=
         "com.microsoft.jdbc.sqlserver.SQLServerDriver";
          protected  static  String  dbUrl="jdbc:microsoft:sqlserver://localhost:
1433;"
                    + "DatabaseName=db_JXC;SelectMethod=Cursor";
          protected static String dbUser="sa";
          protected static String dbPwd="";
          protected static String second=null;
          public static Connection conn=null;
          static {
               try {
                    if (conn==null) {
                         Class.forName(dbClassName).newInstance();
                         conn=DriverManager.getConnection(dbUrl, dbUser,
dbPwd); }
               } catch (Exception ee) {
                    ee.printStackTrace();}     }
          private Dao() {
          // 读取所有客户信息
          public static List getKhInfos() {
               List list=findForList("select id,khname from tb_khinfo");
               return list;}
          // 读取所有供应商信息
          public static List getGysInfos() {
               List list=findForList("select id,name from tb_gysinfo");
               return list;    }
          // 读取客户信息
          public static TbKhinfo getKhInfo(Item item) {
```

```
        String where="khname='"+item.getName()+"'";
        if (item.getId() !=null)
            where="id='" + item.getId()+"'";
        TbKhinfo info=new TbKhinfo();
        ResultSet set=findForResultSet("select*from tb_khinfo where "
                +where);
        try {
            if (set.next()) {
                info.setId(set.getString("id").trim());
                info.setKhname(set.getString("khname").trim());
                info.setJian(set.getString("jian").trim());
                info.setAddress(set.getString("address").trim());
                info.setBianma(set.getString("bianma").trim());
                info.setFax(set.getString("fax").trim());
                info.setHao(set.getString("hao").trim());
                info.setLian(set.getString("lian").trim());
                info.setLtel(set.getString("ltel").trim());
                info.setMail(set.getString("mail").trim());
                info.setTel(set.getString("tel").trim());
                info.setXinhang(set.getString("xinhang").trim());
            }
        } catch (SQLException e) {
            e.printStackTrace();}
        return info;}
    // 读取指定供应商信息
    public static TbGysinfo getGysInfo(Item item) {
        String where="name='"+item.getName()+"'";
        if (item.getId() !=null)
            where="id='" + item.getId()+"'";
        TbGysinfo info=new TbGysinfo();
        ResultSet set=findForResultSet("select*from tb_gysinfo where "
                +where);
        try {
            if (set.next()) {
                info.setId(set.getString("id").trim());
                info.setAddress(set.getString("address").trim());
                info.setBianma(set.getString("bianma").trim());
                info.setFax(set.getString("fax").trim());
                info.setJc(set.getString("jc").trim());
                info.setLian(set.getString("lian").trim());
                info.setLtel(set.getString("ltel").trim());
                info.setMail(set.getString("mail").trim());
                info.setName(set.getString("name").trim());
                info.setTel(set.getString("tel").trim());
                info.setYh(set.getString("yh").trim());}
        } catch (SQLException e) {
```

```
                e.printStackTrace();
            }
            return info;    }
        // 读取用户
        public static TbUserlist getUser(String name, String password) {
            TbUserlist user=new TbUserlist();
            ResultSet  rs=findForResultSet("select*from  tb_userlist  where
name='"
                    +name + "'");
            try {
                if (rs.next()) {
                    user.setName(name);
                    user.setPass(rs.getString("pass"));
                    if (user.getPass().equals(password)) {
                        user.setUsername(rs.getString("username"));
                        user.setQuan(rs.getString("quan"));
                    }}
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return user;}
        // 执行指定查询
        public static ResultSet query(String QueryStr) {
            ResultSet set=findForResultSet(QueryStr);
            return set;     }
        // 执行删除
        public static int delete(String sql) {
            return update(sql);  }
        // 添加客户信息的方法
        public static boolean addKeHu(TbKhinfo khinfo) {
            if (khinfo==null)
                return false;
            return insert("insert tb_khinfo values('" + khinfo.getId()+"','"
                  +khinfo.getKhname()+"','" + khinfo.getJian()+"','"
                  +khinfo.getAddress()+"','" + khinfo.getBianma()+"','"
                  +khinfo.getTel()+"','" + khinfo.getFax()+"','"
                  +khinfo.getLian()+"','" + khinfo.getLtel()+"','"
                  +khinfo.getMail()+"','" + khinfo.getXinhang()+"','"
                  +khinfo.getHao()+"')"); }
        // 修改客户信息的方法
        public static int updateKeHu(TbKhinfo khinfo) {
            return update("update tb_khinfo set jian='"+khinfo.getJian()
                    +"',address='"+khinfo.getAddress()+"',bianma='"
                    +khinfo.getBianma()+"',tel='"+khinfo.getTel()+"',fax='"
                    +khinfo.getFax()+"',lian='"+khinfo.getLian()+"',ltel='"
                    +khinfo.getLtel()+"',mail='"+khinfo.getMail()
```

```
                +"',xinhang='"+khinfo.getXinhang()+"',hao='"
                +khinfo.getHao()+"'where id='"+khinfo.getId()+"'");  }
        // 修改库存的方法
        public static int updateKucunDj(TbKucun kcInfo) {
            return update("update tb_kucun set dj="+kcInfo.getDj()
                +" where id='"+kcInfo.getId()+"'");   }
        // 修改供应商信息的方法
        public static int updateGys(TbGysinfo gysInfo) {
            return update("update tb_gysinfo set jc='"+gysInfo.getJc()
                +"',address='"+gysInfo.getAddress()+"',bianma='"
                +gysInfo.getBianma()+"',tel='"+gysInfo.getTel ()
                +"',fax='"+gysInfo.getFax()+"',lian='"+gysInfo.  get
Lian()
                +"',ltel='"+gysInfo.getLtel()+"',mail='"
                +gysInfo.getMail()+"',yh='"+gysInfo.getYh()
                +"' where id='"+gysInfo.getId() + "'");}
        // 添加供应商信息的方法
        public static boolean addGys(TbGysinfo gysInfo) {
            if (gysInfo==null)
                return false;
            return insert("insert tb_gysinfo values('"+gysInfo.getId()+"','"
                +gysInfo.getName()+"','"+gysInfo.getJc()+"','"
                +gysInfo.getAddress()+"','"+gysInfo.getBianma() + "','"
                +gysInfo.getTel()+"','"+gysInfo.getFax()+"','"
                +gysInfo.getLian()+"','"+gysInfo.getLtel()+"','"
                +gysInfo.getMail()+"','"+gysInfo.getYh()+"')");  }
        // 添加商品
        public static boolean addSp(TbSpinfo spInfo) {
            if (spInfo==null)
                return false;
            return insert("insert tb_spinfo values('"+spInfo.getId()+"','"
                +spInfo.getSpname()+"','"+spInfo.getJc()+"','"
                +spInfo.getCd()+"','"+spInfo.getDw()+"','"
                +spInfo.getGg()+"','"+spInfo.getBz()+"','"
                +spInfo.getPh()+"','"+spInfo.getPzwh()+"','"
                +spInfo.getMemo()+"','"+spInfo.getGysname()+"')");}
        // 更新商品
        public static int updateSp(TbSpinfo spInfo) {
            return update("update tb_spinfo set jc='"+spInfo.getJc()+"',cd='"
                +spInfo.getCd()+"',dw='"+spInfo.getDw()+"',gg='"
                +spInfo.getGg()+"',bz='"+spInfo.getBz()+"',ph='"
                +spInfo.getPh()+"',pzwh='"+spInfo.getPzwh()+"',memo='"
                +spInfo.getMemo()+"',gysname='"+spInfo.getGysname()
                +"'where id='"+spInfo.getId()+"'");}
        // 读取商品信息
```

```
    public static TbSpinfo getSpInfo(Item item) {
        String where="spname='"+item.getName()+"'";
        if (item.getId() !=null)
            where="id='"+item.getId()+"'";
        ResultSet rs=findForResultSet("select * from tb_spinfo where "
                +where);
        TbSpinfo spInfo=new TbSpinfo();
        try {
            if (rs.next()) {
                spInfo.setId(rs.getString("id").trim());
                spInfo.setBz(rs.getString("bz").trim());
                spInfo.setCd(rs.getString("cd").trim());
                spInfo.setDw(rs.getString("dw").trim());
                spInfo.setGg(rs.getString("gg").trim());
                spInfo.setGysname(rs.getString("gysname").trim());
                spInfo.setJc(rs.getString("jc").trim());
                spInfo.setMemo(rs.getString("memo").trim());
                spInfo.setPh(rs.getString("ph").trim());
                spInfo.setPzwh(rs.getString("pzwh").trim());
                spInfo.setSpname(rs.getString("spname").trim());}
        } catch (SQLException e) {
            e.printStackTrace();}
        return spInfo;}
    // 获取所有商品信息
    public static List getSpInfos() {
        List list=findForList("select*from tb_spinfo");
        return list;}
    // 获取库存商品信息
    public static TbKucun getKucun(Item item) {
        String where="spname='"+item.getName()+"'";
        if (item.getId() !=null)
            where="id='" + item.getId() + "'";
        ResultSet rs=findForResultSet("select*from tb_kucun where " +
where);
        TbKucun kucun=new TbKucun();
        try {
            if (rs.next()) {
                kucun.setId(rs.getString("id"));
                kucun.setSpname(rs.getString("spname"));
                kucun.setJc(rs.getString("jc"));
                kucun.setBz(rs.getString("bz"));
                kucun.setCd(rs.getString("cd"));
                kucun.setDj(rs.getDouble("dj"));
                kucun.setDw(rs.getString("dw"));
                kucun.setGg(rs.getString("gg"));
                kucun.setKcsl(rs.getInt("kcsl"));}
```

```
            } catch (SQLException e) {
                e.printStackTrace();}
            return kucun;}
      // 获取入库单的最大 ID，即最大入库票号
      public static String getRuKuMainMaxId(Date date) {
        return getMainTypeTableMaxId(date, "tb_ruku_main", "RK", "rkid");}
        // 在事务中添加入库信息
        public static boolean insertRukuInfo(TbRukuMain ruMain) {
            try {
                boolean autoCommit = conn.getAutoCommit();
                conn.setAutoCommit(false);
                // 添加入库主表记录
               insert("insert into tb_ruku_main values('" + ruMain.getRkId()
                        + "','" + ruMain.getPzs() + "'," + ruMain.getJe()
+ ",'"
                        + ruMain.getYsjl() + "','" + ruMain.getGysname()
+ "','"
                        + ruMain.getRkdate() + "','" + ruMain.getCzy() +
"','"
                        +  ruMain.getJsr()  +  "','"  +  ruMain.getJsfs()  +
"')");
               Set<TbRukuDetail>  rkDetails  =  ruMain.getTabRukuDetails();
                for (Iterator<TbRukuDetail> iter = rkDetails.iterator(); iter
                        .hasNext();) {
                    TbRukuDetail details = iter.next();
                    // 添加入库详细表记录
                    insert("insert  into  tb_ruku_detail  values('"  +  ruMain.
getRkId()
                            + "','" + details.getTabSpinfo() + "',"
                            + details.getDj() + "," + details.getSl() +
")");
                    // 添加或修改库存表记录
                    Item item = new Item();
                    item.setId(details.getTabSpinfo());
                    TbSpinfo spInfo = getSpInfo(item);
                    if (spInfo.getId() != null && !spInfo.getId().isEmpty())
{
                        TbKucun kucun = getKucun(item);
                        if (kucun.getId() == null || kucun.getId().isEmpty())
{
                            insert("insert into tb_kucun values('" + spInfo.
getId()
                                    +  "','"  +  spInfo.getSpname()  +
"','"
                                    + spInfo.getJc() + "','" + spInfo.
getCd()
```

```
                                        + "','" + spInfo.getGg() + "','"
                                        + spInfo.getBz() + "','" + spInfo.
getDw()
                                        + "'," + details.getDj() + ","
                                        + details.getSl() + ")");
                        } else {
                            int sl = kucun.getKcsl() + details.getSl();
                            update("update tb_kucun set kcsl=" + sl + ",dj="
                                        + details.getDj() + " where id='"
                                        + kucun.getId() + "'");
                        }    }}
                conn.commit();
                conn.setAutoCommit(autoCommit);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return true;
        }
        public static ResultSet findForResultSet(String sql) {
            if (conn == null)
                return null;
            long time = System.currentTimeMillis();
            ResultSet rs = null;
            try {
                Statement stmt = null;
               stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                        ResultSet.CONCUR_READ_ONLY);
                rs = stmt.executeQuery(sql);
               second = ((System.currentTimeMillis() - time) / 1000d) + "";
            } catch (Exception e) {
                e.printStackTrace();
            }
            return rs;
        }
        public static boolean insert(String sql) {
            boolean result = false;
            try {
                Statement stmt = conn.createStatement();
                result = stmt.execute(sql);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return result;
        }
        public static int update(String sql) {
            int result = 0;
```

```
				try {
					Statement stmt = conn.createStatement();
					result = stmt.executeUpdate(sql);
				} catch (SQLException e) {
					e.printStackTrace();
				}
				return result;
			}
		public static List findForList(String sql) {
			List<List> list = new ArrayList<List>();
			ResultSet rs = findForResultSet(sql);
			try {
				ResultSetMetaData metaData = rs.getMetaData();
				int colCount = metaData.getColumnCount();
				while (rs.next()) {
					List<String> row = new ArrayList<String>();
					for (int i = 1; i <= colCount; i++) {
						String str = rs.getString(i);
						if (str != null && !str.isEmpty())
							str = str.trim();
						row.add(str);
					}
					list.add(row);
				}
			} catch (Exception e) {
				e.printStackTrace();
			}
			return list;
		}
			// 获取退货最大 ID
			public static String getRkthMainMaxId(Date date) {
				return getMainTypeTableMaxId(date, "tb_rkth_main", "RT", "rkthId");
			}
			// 在事务中添加入库退货信息
			public static boolean insertRkthInfo(TbRkthMain rkthMain) {
				try {
					boolean autoCommit = conn.getAutoCommit();
					conn.setAutoCommit(false);
					// 添加入库退货主表记录
					insert("insert into tb_rkth_main values('" + rkthMain.getRkthId()
							+ "','" + rkthMain.getPzs() + "'," + rkthMain.getJe()
							+ ",'" + rkthMain.getYsjl() + "','" + rkthMain. get
Gysname()
							+ "','" + rkthMain.getRtdate() + "','" + rkthMain.
getCzy()
							+ "','" + rkthMain.getJsr() + "','" + rkthMain. get
```

```
Jsfs()
                                + "')");
                    Set<TbRkthDetail> rkDetails = rkthMain.getTbRkthDetails();
                    for (Iterator<TbRkthDetail> iter = rkDetails.iterator(); iter
                                .hasNext();) {
                         TbRkthDetail details = iter.next();
                         // 添加入库详细表记录
                         insert("insert into tb_rkth_detail values('"
                                     + rkthMain.getRkthId() + "','" + details. Get
Spid()
                                     + "'," + details.getDj() + "," + details. get
Sl() + ")");
                         // 添加或修改库存表记录
                         Item item = new Item();
                         item.setId(details.getSpid());
                         TbSpinfo spInfo = getSpInfo(item);
                         if  (spInfo.getId()  !=  null  &&  !spInfo.getId().  Is
Emp ty()) {
                              TbKucun kucun = getKucun(item);
                              if (kucun.getId() != null && !kucun.getId(). Is
Empty()) {
                                   int sl = kucun.getKcsl() - details. getSl();
                                   update("update tb_kucun set kcsl=" + sl + "
where id='"
                                             + kucun.getId() + "'");
                              }
                         }
                    }
                    conn.commit();
                    conn.setAutoCommit(autoCommit);
               } catch (SQLException e) {
                    e.printStackTrace();
               }
               return true;
          }
          // 获取销售主表最大 ID
          public static String getSellMainMaxId(Date date) {
               return getMainTypeTableMaxId(date, "tb_sell_main", "XS", "sellID");
          }
          // 在事务中添加销售信息
          public static boolean insertSellInfo(TbSellMain sellMain) {
               try {
                    boolean autoCommit = conn.getAutoCommit();
                    conn.setAutoCommit(false);
                    // 添加销售主表记录
                   insert("insert into tb_sell_main values('" + sellMain.getSellId()
```

```
                        + "','" + sellMain.getPzs() + "','" + sellMain.getJe()
                         + "','" + sellMain.getYsjl() + "','" + sellMain. Get
Khname()
                         + "','" + sellMain.getXsdate() + "','" + sellMain.
getCzy()
                         + "','" + sellMain.getJsr() + "','" + sellMain. get
Jsfs()
                         + "')");
                Set<TbSellDetail> rkDetails = sellMain.getTbSellDetails();
                for (Iterator<TbSellDetail> iter = rkDetails.iterator(); iter
                        .hasNext();) {
                    TbSellDetail details = iter.next();
                    // 添加销售详细表记录
                    insert("insert into tb_sell_detail values('"
                            + sellMain.getSellId() + "','" + details. get
Spid()
                            + "'," + details.getDj() + "," + details. get
Sl() + ")");
                    // 修改库存表记录
                    Item item = new Item();
                    item.setId(details.getSpid());
                    TbSpinfo spInfo = getSpInfo(item);
                    if (spInfo.getId() != null && !spInfo.getId().isEmpty()) {
                        TbKucun kucun = getKucun(item);
                        if (kucun.getId() != null && !kucun.getId().isEmpty())
{
                            int sl = kucun.getKcsl() - details.getSl();
                            update("update  tb_kucun  set  kcsl=" + sl + "
where id='"
                                    + kucun.getId() + "'");
                        }    }}
                conn.commit();
                conn.setAutoCommit(autoCommit);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return true;
        }
        // 获取更类主表最大 ID
       private static String getMainTypeTableMaxId(Date date, String table,
                String idChar, String idName) {
            String dateStr = date.toString().replace("-", "");
            String id = idChar + dateStr;
            String sql = "select max(" + idName + ") from " + table + " where "
                    + idName + " like '" + id + "%'";
            ResultSet set = query(sql);
```

```
            String baseId = null;
            try {
                if (set.next())
                    baseId = set.getString(1);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            baseId = baseId == null ? "000" : baseId.substring(baseId.length()-
3);
            int idNum = Integer.parseInt(baseId) + 1;
            id += String.format("%03d", idNum);
            return id;
        }
        public static String getXsthMainMaxId(Date date) {
            return getMainTypeTableMaxId(date, "tb_xsth_main", "XT", "xsthID");
        }
        public static List getKucunInfos() {
            List list = findForList("select id,spname,dj,kcsl from tb_kucun");
            return list;
        }
        // 在事务中添加销售退货信息
        public static boolean insertXsthInfo(TbXsthMain xsthMain) {
            try {
                boolean autoCommit = conn.getAutoCommit();
                conn.setAutoCommit(false);
                // 添加销售退货主表记录
                insert("insert into tb_xsth_main values('" + xsthMain.getXsthId()
                        + "','" + xsthMain.getPzs() + "'," + xsthMain.getJe()
                        + ",'" + xsthMain.getYsjl() + "','" + xsthMain.
getKhname()
                        + "','" + xsthMain.getThdate() + "','" + xsthMain.
getCzy()
                        + "','" + xsthMain.getJsr() + "','" + xsthMain.
getJsfs()
                        + "')");
                Set<TbXsthDetail> xsthDetails = xsthMain.getTbXsthDetails();
                for (Iterator<TbXsthDetail> iter = xsthDetails.iterator(); iter
                        .hasNext();) {
                    TbXsthDetail details = iter.next();
                    // 添加销售退货详细表记录
                    insert("insert into tb_xsth_detail values('"
                            + xsthMain.getXsthId() + "','" + details.get
Spid()
                            + "'," + details.getDj() + "," + details.
getSl()+ ")");
                    // 修改库存表记录
```

```
                        Item item = new Item();
                        item.setId(details.getSpid());
                        TbSpinfo spInfo = getSpInfo(item);
                        if (spInfo.getId() != null && !spInfo.getId().isEmpty()) {
                             TbKucun kucun = getKucun(item);
                            if (kucun.getId() != null && !kucun.getId().isEmpty())
{
                                  int sl = kucun.getKcsl() + details.getSl();
                                  update("update tb_kucun set kcsl=" + sl + "
where id='"
                                            + kucun.getId() + "'");
                             }
                        }
                   }
                   conn.commit();
                   conn.setAutoCommit(autoCommit);
              } catch (SQLException e) {
                   e.printStackTrace();
              }
              return true;
         }
         // 添加用户
         public static int addUser(TbUserlist ul) {
              return update("insert tb_userlist values('" + ul.getUsername() + "','"
                       + ul.getName() + "','" + ul.getPass() + "','" + ul.getQuan()
                        + "')");
         }
         public static List getUsers() {
              List list = findForList("select * from tb_userlist");
              return list;
         }
         // 修改用户方法
         public static int updateUser(TbUserlist user) {
              return update("update tb_userlist set username='" + user.getUsername()
                        + "',name='" + user.getName() + "',pass='" + user.getPass()
                        + "',quan='" + user.getQuan() + "' where name='"
                        + user.getName() + "'");
         }
         // 获取用户对象的方法
         public static TbUserlist getUser(Item item) {
              String where = "name='" + item.getName() + "'";
              if (item.getId() != null)
                   where = "username='" + item.getId() + "'";
              ResultSet rs = findForResultSet("select * from tb_userlist where "
                        + where);
              TbUserlist user=new TbUserlist();
```

```
            try {
                if (rs.next()) {
                    user.setName(rs.getString("name").trim());
                    user.setUsername(rs.getString("username").trim());
                    user.setPass(rs.getString("pass").trim());
                    user.setQuan(rs.getString("quan").trim());
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return user;
        }
    }
```

温故而知新——第四篇实战范例

第四篇是实例篇，综合前面所学的知识在最后一章编写了一个数据库管理系统。本节将对全篇的知识和技术要点进行回顾，使读者对这些知识的理解得到升华。

范例 1　编写记事本

在本篇第一章编写了一个画图板，下面展示一段编写记事本的代码，其核心思想和编写画图板类似，只是所用的方法略有不同。代码如下：

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
class TextEditorFrame extends JFrame{
   File file=null;
   Color color=Color.red;
      TextEditorFrame(){
   initTextPane();
   initAboutDialog();
   initToolBar();
   initMenu();
         }
   void initTextPane(){
    getContentPane().add(new JScrollPane(text));
    }
       JTextPane text=new JTextPane(); //这是用来做文本框的
    JFileChooser filechooser=new JFileChooser(); //文件选择框
    JColorChooser colorchooser=new JColorChooser();//
    JDialog about=new JDialog(this); //关于对话框
    JMenuBar menubar=new JMenuBar();//菜单
    JMenu[] menus=new JMenu[]{
 new JMenu("文件"),
 new JMenu("编辑"),
 new JMenu("帮助")
 };
JMenuItem menuitems[][]=new JMenuItem[][]{{
 new JMenuItem("新建"),
 new JMenuItem("打开"),
```

```
            new JMenuItem("保存"),
            new JMenuItem("退出")
                   },

             {
            new JMenuItem("复制"),
            new JMenuItem("剪切"),
            new JMenuItem("粘贴"),
            new JMenuItem("颜色")
                  },
                   {
                  new JMenuItem("关于")
                   }
                  };
                void initMenu(){
                        for(int i=0;i<menus.length;i++){
                    menubar.add(menus[i]);
                    for(int j=0;j<menuitems[i].length;j++){
                     menus[i].add(menuitems[i][j]);
                     menuitems[i][j].addActionListener( action );
                    }
                   }
                   this.setJMenuBar(menubar);
                }
               ActionListener action=new ActionListener(){
               public void actionPerformed(ActionEvent e){
            JMenuItem mi=(JMenuItem)e.getSource();
            String id=mi.getText();
            if(id.equals("新建")){
             text.setText("");
             file=null;
            }else if(id.equals("打开")){
                              if(file !=null)filechooser.setSelectedFile(file);
                              int returnVal=filechooser.showOpenDialog(TextEditor
Frame.this);
                              if(returnVal==JFileChooser.APPROVE_OPTION){
                              file=filechooser.getSelectedFile();
                              openFile();
                                        }
                }else if(id.equals("保存")){
                 if(file!=null) filechooser.setSelectedFile(file);
                    int returnVal=filechooser.showSaveDialog(TextEditorFrame.this);
                          if(returnVal==JFileChooser.APPROVE_OPTION){
                          file=filechooser.getSelectedFile();
                            saveFile();
                                            }
```

```
                            }else if(id.equals("退出")){
                        TextEditorFrame f=new TextEditorFrame();
                        int s=JOptionPane.showConfirmDialog(f,"你真的要退出吗","退出
程序",JOptionPane.YES_NO_CANCEL_OPTION);
                        if(s==JOptionPane.YES_OPTION)
                         System.exit(0);
                      }else if(id.equals("剪切")){
                        text.cut();
                      }else if(id.equals("复制")){
                        text.copy();
                      }else if(id.equals("粘贴")){
                        text.paste();
                      }else if(id.equals("color")){
                      color=JColorChooser.showDialog(TextEditorFrame.this,"",
color);
                          text.setForeground(color);

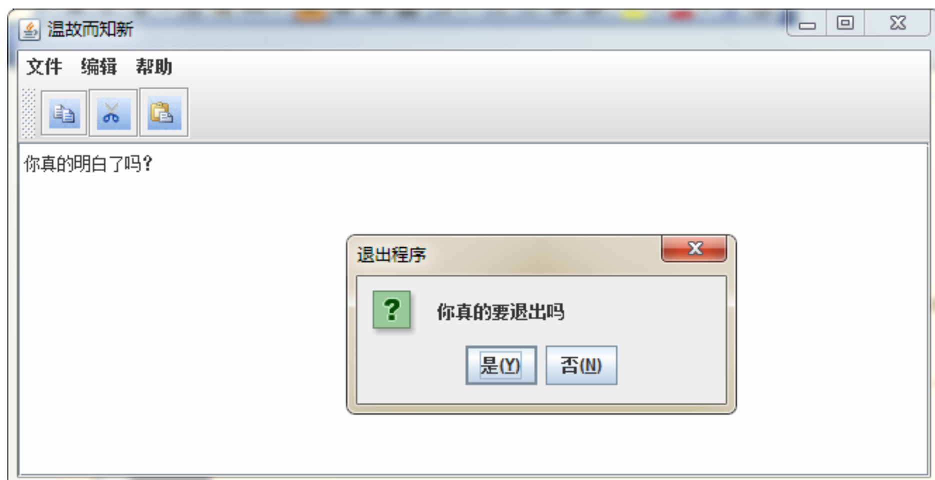
             }else if(id.equals("关于")){
                       about.setSize(200,150);
                       about.show();
              }
                }
           };
           void saveFile(){
                try{
                 FileWriter fw=new FileWriter(file);
                 fw.write(text.getText());
                 fw.close();
                  }
             catch(Exception e){e.printStackTrace();}
                            }
            void openFile(){
                  try{
                       FileReader fr=new FileReader(file);
                       int len=(int)file.length();
                       char []buffer=new char[len];
                       fr.read(buffer,0,len);
                       fr.close();
                       text.setText(new String(buffer));
                       }catch(Exception e){e.printStackTrace();}
                }
            void initAboutDialog(){
                           about.getContentPane().add(new JLabel("传到 Java，你进步了
吗？"));
                           about.setModal(true);
                           about.setSize(200,100);
```

```
            }
          JToolBar toolbar=new JToolBar();//我来加上工具条
              JButton [] buttons=new JButton[]{
               new JButton("", new ImageIcon("qin.jpg")),
               new JButton("", new ImageIcon("shu.jpg")),
               new JButton("", new ImageIcon("xin.jpg"))
            };
         void initToolBar(){
               for(int i=0;i<buttons.length;i++)
               toolbar.add(buttons[i]);
              buttons[0].setToolTipText("复制");
              buttons[0].addActionListener(new ActionListener(){
              public void actionPerformed(ActionEvent e){
                   text.copy();
            }
         });
           buttons[1].setToolTipText("剪切");
           buttons[1].addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                  text.cut();
            }
         });
                buttons[2].setToolTipText("粘贴");
                buttons[2].addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e){
                  text.paste();
            }
         });
              this.getContentPane().add(toolbar,BorderLayout.NORTH);
           }
       }
         public class F{
       public static void main(String args[]){
                     TextEditorFrame f=new TextEditorFrame();

                  f.addWindowListener(new WindowAdapter(){
                      public void windowClosing(WindowEvent e)
                    {
                   TextEditorFrame f=new TextEditorFrame();
                  int s=JOptionPane.showConfirmDialog(f,"你真的要退出吗","退出
程序",JOptionPane.YES_NO_OPTION);
                  if(s==JOptionPane.YES_OPTION)
                   System.exit(0);}
                });
                    f.setTitle("温故而知新");
        f.setSize(800,600);
```

```
        f.show();
       }
        }
```

运行代码，得到如范例图 4-1 所示的结果。



范例图 4-1　运行结果

范例 2　使用 Java 编写简易计算器

使用 Java 编写计算器需要清楚加减乘除运算的规则，通过简易计算器的编写，初学者可以更深入地理解监听、窗口中组件功能的编程，下面展示简易计算器编写方法，其代码如下：

```
        //声明需要插入的包
        import java.awt.*;
        import java.lang.Object;
            import java.lang.String;
        import javax.swing.*;
        import java.awt.event.*;
        import java.awt.TextComponent;
        //声明一个主类 jisuanqi，继续窗口类 Frame
        public class jisuanqi extends Frame{
        //声明 16 个 Button
        Button  anjian1,  anjian2,  anjian3,  anjian4,  anjian5,  anjian6,
anjian7, anjian8, anjian9, anjian10, anjian11, anjian12, anjian13, anjian14,
anjian15,anjian16;
        //声明文本框，用于接收数字和计算
        TextField numText;
        //声明面板容器，用于存放各种按钮
        Panel anjianPanel;
        //构造方法初始化按钮
```

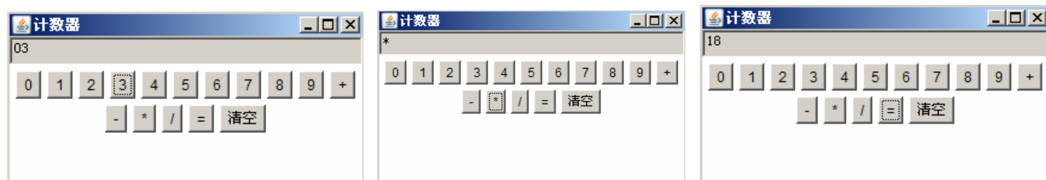
```
jisuanqi(String title){
super(title);
anjian1=new Button("0");
anjian2=new Button("1");
anjian3=new Button("2");
anjian4=new Button("3");
anjian5=new Button("4");
anjian6=new Button("5");
anjian7=new Button("6");
anjian8=new Button("7");
anjian9=new Button("8");
anjian10=new Button("9");
anjian11=new Button("+");
anjian12=new Button("-");
anjian13=new Button("*");
anjian14=new Button("/");
anjian15=new Button("=");
anjian16=new Button("清空");
//初始化文本框，将文本的值设置为“0”
numText=new TextField("0");
//实例化面板;
anjianPanel=new Panel();
//设置按钮
anjianPanel.setLayout(new FlowLayout());
//实例化按钮
anjian1.addActionListener(new anjianAction());
anjian2.addActionListener(new anjianAction());
anjian3.addActionListener(new anjianAction());
anjian4.addActionListener(new anjianAction());
anjian5.addActionListener(new anjianAction());
anjian6.addActionListener(new anjianAction());
anjian7.addActionListener(new anjianAction());
anjian8.addActionListener(new anjianAction());
anjian9.addActionListener(new anjianAction());
anjian10.addActionListener(new anjianAction());
anjian11.addActionListener(new anjianAction());
anjian12.addActionListener(new anjianAction());
anjian13.addActionListener(new anjianAction());
anjian14.addActionListener(new anjianAction());
anjian15.addActionListener(new anjianAction());
anjian16.addActionListener(new anjianAction());
//将各个按钮添加到面板容器中
anjianPanel.add(anjian1);
anjianPanel.add(anjian2);
anjianPanel.add(anjian3);
anjianPanel.add(anjian4);
```

```
anjianPanel.add(anjian5);
anjianPanel.add(anjian6);
anjianPanel.add(anjian7);
anjianPanel.add(anjian8);
anjianPanel.add(anjian9);
anjianPanel.add(anjian10);
anjianPanel.add(anjian11);
anjianPanel.add(anjian12);
anjianPanel.add(anjian13);
anjianPanel.add(anjian14);
anjianPanel.add(anjian15);
anjianPanel.add(anjian16);
//为文本框添加事件
numText.addTextListener(new TextListener() {
public void textValueChanged(TextEvent e) {
if(numText.getText().indexOf("0",0)!=-1){
numText.getText().replace("0","");
}
}
});
addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e) {
System.exit(0);
}
});
//将文本框放置在顶部
add(numText,BorderLayout.NORTH);
//添加按钮
add(anjianPanel,BorderLayout.CENTER);
//设置文本框属性
numText.setEditable(false);
}
//没有任何操作的值
int num=0;
//numStr 是文本框的值
String numStr=null;
Boolean chinage=false;
//设置按钮
public int check(){
numStr=numText.getText();
if (numStr.indexOf("+")==0) {
numStr=numStr.replace('+', '0');
num=num + Integer.parseInt(numStr);
return num;
}else
if (numStr.indexOf("-") !=-1) {
```

```
numStr=numStr.replace("-", "0");
num=num-Integer.parseInt(numStr);
return num;
}
else
if(numStr.indexOf("*")!=-1){
numStr=numStr.replace('*','0');
num=num * Integer.parseInt(numStr);
return num;
}
else
if(numStr.indexOf("/")!=-1){
numStr=numStr.replace('/','0');
try{
num=num / Integer.parseInt(numStr);
return num;
}catch(ArithmeticException e){
JOptionPane.showMessageDialog(null,"除数不能为空!","消息!",1);
return num;}}
else
return num=Integer.parseInt(numStr);}
//创建事件
private class anjianAction implements ActionListener{
public void actionPerformed(ActionEvent event){
if(event.getActionCommand()=="0"){
if(!numText.getText().equals("0"))
numText.setText(numText.getText()+0);}
if(event.getActionCommand()=="1")
numText.setText(numText.getText()+1);
if(event.getActionCommand()=="2")
numText.setText(numText.getText()+2);
if(event.getActionCommand()=="3")
numText.setText(numText.getText()+3);
if(event.getActionCommand()=="4")
numText.setText(numText.getText()+4);
if(event.getActionCommand()=="5")
numText.setText(numText.getText()+5);
if(event.getActionCommand()=="6")
numText.setText(numText.getText()+6);
if(event.getActionCommand()=="7")
numText.setText(numText.getText()+7);
if(event.getActionCommand()=="8")
numText.setText(numText.getText()+8);
if(event.getActionCommand()=="9")
numText.setText(numText.getText()+9);
if(event.getActionCommand()=="清空"){
```

```
        numText.setText("0");
        num=0;}
        if(event.getActionCommand()=="+"){
        //调用 check();方法
        check();
        numText.setText("+");}
        if(event.getActionCommand()=="-"){
        check();
        numText.setText("-");}
        if(event.getActionCommand()=="*"){
        check();
        numText.setText("*");}
        if(event.getActionCommand()=="/"){
        if(numStr=="/0"){
        JOptionPane.showMessageDialog(null,"除数不能为空!","消息!",1);}
        check();
        numText.setText("/");}
        if(event.getActionCommand()=="="){
        check();
        numText.setText(""+num);}}}
        public static void main(String[] args){
        jisuanqi jisuanqi=new jisuanqi("计数器");
        jisuanqi.setSize(300,280);
        //设置窗口大小
        jisuanqi.setLocation(550,350);
        //设置界面属性
        jisuanqi.setVisible(true);
        }}
```

执行程序，得到如范例图 4-2 所示的结果。



范例图 4-2　简易计算机运行结果

ISBN 978-7-111-37937-9

◎策划编辑：丁 诚

◎封面设计：道乐 dollor 道乐文化

《C语言编程新手自学手册》
《Java编程新手自学手册》
《Java Web编程新手自学手册》
《PHP编程新手自学手册》
《Visual C++编程新手自学手册》
《C＃编程新手自学手册》
《编程算法新手自学手册》

上架建议：计算机/程序设计

地址：北京市百万庄大街22号
电话服务
社服务中心：(010) 88361066
销 售 一 部：(010) 68326294
销 售 二 部：(010) 88379649
读者购书热线：(010) 88379203

邮政编码：100037
网络服务
门户网：http://www.cmpbook.com
教材网：http://www.cmpedu.com
封面无防伪标均为盗版

定价：89.90元(含1DVD)

